INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN
LEHRSTUHL FÜR EFFIZIENTE ALGORITHMEN

Skriptum

zur Vorlesung

Algorithmische Bioinformatik I/II

gehalten im Wintersemester 2001/2002

und im Sommersemester 2002 von

Volker Heun

Erstellt unter Mithilfe von:

Peter Lücke – Hamed Behrouzi – Michael Engelhardt

Sabine Spreer – Hanjo Täubig

Jens Ernst – Moritz Maaß

14. Mai 2003

Version 0.96

Vorwort

Dieses Skript entstand parallel zu den Vorlesungen *Algorithmische Bioinformatik I* und *Algorithmische Bioinformatik II*, die im Wintersemester 2001/2002 sowie im Sommersemester 2002 für Studenten der Bioinformatik und Informatik sowie anderer Fachrichtungen an der Technischen Universität München im Rahmen des von der Ludwig-Maximilians-Universität und der Technischen Universität gemeinsam veranstalteten Studiengangs Bioinformatik gehalten wurde. Einige Teile des Skripts basieren auf der bereits im Sommersemester 2000 an der Technischen Universität München gehaltenen Vorlesung *Algorithmen der Bioinformatik* für Studierende der Informatik.

Das Skript selbst umfasst im Wesentlichen die grundlegenden Themen, die man im Bereich Algorithmische Bioinformatik einmal gehört haben sollte. Die vorliegende Version bedarf allerdings noch einer Ergänzung weiterer wichtiger Themen, die leider nicht in den Vorlesungen behandelt werden konnten.

An dieser Stelle möchte ich insbesondere Hamed Behrouzi, Michael Engelhardt und Peter Lücke danken, die an der Erstellung des ersten Teils dieses Skriptes (Kapitel 2 mit 5) maßgeblich beteiligt waren. Bei Sabine Spreer möchte ich mich für die Unterstützung bei Teilen des siebten Kapitels bedanken. Bei meinen Übungsleitern Jens Ernst und Moritz Maaß für deren Unterstützung der Durchführung des Übungsbetriebs, aus der einige Lösungen von Übungsaufgaben in dieses Text eingeflossen sind. Bei Hanjo Täubig möchte ich mich für die Mithilfe zur Fehlerfindung bedanken, insbesondere bei den biologischen Grundlagen.

Falls sich dennoch weitere (Tipp)Fehler unserer Aufmerksamkeit entzogen haben sollten, so bin ich für jeden Hinweis darauf (an heun@in.tum.de) dankbar.

München, im September 2002 — Volker Heun

Inhaltsverzeichnis

Phylogenetische Bäume

7.1 Einleitung

In diesem Kapitel wollen wir uns mit *phylogenetischen Bäumen* bzw. *evolutionären Bäumen* beschäftigen. Wir wollen also die Entwicklungsgeschichte mehrerer verwandter Spezies anschaulich als Baum darstellen bzw. das Auftreten von Unterschieden in den Spezies durch Verzweigungen in einem Baum wiedergeben.

Definition 7.1 *Ein* phylogenetischer Baum *für eine Menge* $S = \{s_1, \ldots, s_n\}$ *von* n *Spezies ist ein ungeordneter gewurzelter Baum mit* n *Blättern und den folgenden Eigenschaften:*

- *Jeder innere Knoten hat mindestens zwei Kinder;*
- *Jedes Blatt ist mit genau einer Spezies* $s \in S$ *markiert;*
- *Jede Spezies taucht nur einmal als Blattmarkierung auf.*

Ungeordnet bedeutet hier, dass die Reihenfolge der Kinder eines Knotens ohne Belang ist. Die bekannten und noch lebenden (zum Teil auch bereits ausgestorbenen) Spezies werden dabei an den Blättern dargestellt. Jeder (der maximal $n-1$) inneren Knoten entspricht dann einem Ahnen der Spezies, die in seinem Teilbaum die Blätter bilden. In Abbildung 7.1 ist ein Beispiel eines phylogenetischen Baumes angegeben.

Wir wollen uns hier mit der mathematischen und algorithmischen Rekonstruktion von phylogenetischen Bäumen anhand der gegebenen biologischen Daten beschäftigen. Die daraus resultierenden Bäume müssen daher nicht immer mit der biologischen Wirklichkeit übereinstimmen. Die rekonstruierten phylogenetischen Bäume mögen Ahnen vorhersagen, die niemals existiert haben.

Dies liegt zum einen daran, dass die biologischen Daten nicht in der Vollständigkeit und Genauigkeit vorliegen, die für eine mathematische Rekonstruktion nötig sind. Zum anderen liegt dies auch an den vereinfachenden Modellen, da in der Natur nicht nur Unterscheidungen (d.h. Verzweigungen in den Bäumen) vorkommen können, sondern dass auch Vereinigungen bereits getrennter Spezies vorkommen können.

Biologisch würde man daher eher nach einem gerichteten azyklischen Graphen statt eines gewurzelten Baumes suchen. Da diese Verschmelzungen aber eher vereinzelt

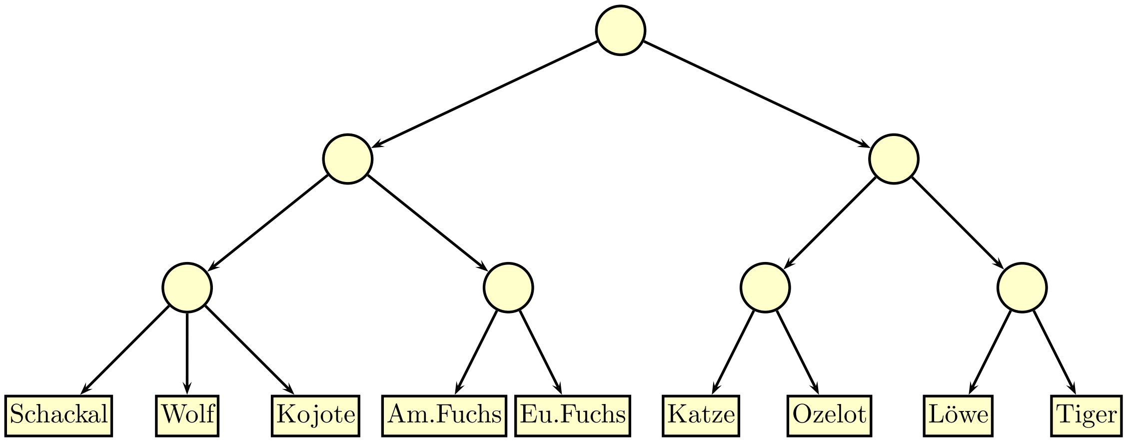


Abbildung 7.1: Beispiel: Ein phylogenetischer Baum

vorkommen, bilden phylogenetischer Bäume einen ersten Ansatzpunkt, in die dann weiteres biologisches Wissen eingearbeitet werden kann.

In der Rekonstruktion unterscheidet man zwei prinzipiell unterschiedliche Verfahren: distanzbasierte und charakterbasierte Verfahren, die wir in den beiden folgenden Unterabschnitten genauer erörtern werden.

7.1.1 Distanzbasierte Verfahren

Bei den so genannten *distanzbasierten Verfahren* wird zwischen den Spezies ein Abstand bestimmt. Man kann diesen einfach als die Zeitspanne in die Vergangenheit interpretieren, vor der sich die beiden Spezies durch Spezifizierung aus einem gemeinsamen Urahn auseinander entwickelt haben.

Für solche Distanzen, also evolutionäre Abstände, können beispielsweise die EDIT-Distanzen von speziellen DNS-Teilsträngen oder Aminosäuresequenzen verwendet werden. Hierbei wird angenommen, dass durch Mutationen die Sequenzen sich auseinander entwickeln und dass die Anzahl der so genannten akzeptierten Mutationen (also derer, die einem Weiterbestehen der Art nicht im Wege standen) zur zeitlichen Dauer korreliert ist. Hierbei muss man vorsichtig sein, da unterschiedliche Bereiche im Genom auch unterschiedliche Mutationsraten besitzen.

Eine andere Möglichkeit aus früheren Tagen sind Hybridisierungsexperimente. Dabei werden durch vorsichtiges Erhitzen die DNS-Doppelstränge zweier Spezies voneinander getrennt. Bei der anschließenden Abkühlung hybridisieren die DNS-Stränge wieder miteinander. Da jetzt jedoch DNS-Einzelstränge von zwei Spezies vorliegen, können auch zwei Einzelstränge von zwei verschiedenen Spezies miteinander hybridisieren, vorausgesetzt, die Stränge waren nicht zu verschieden.

Beim anschließenden erneuten Erhitzen trennen sich diese gemischten Doppelstränge umso schneller, je verschiedener die DNS-Sequenzen sind, da dann entsprechend weniger Wasserstoffbrücken aufzubrechen sind. Aus den Temperaturen, bei denen sich dann diese gemischten DNS-Doppelstränge wieder trennen, kann man dann ein evolutionäres Abstandsmaß gewinnen.

Ziel der evolutionären Verfahren ist es nun, einen Baum mit Kantengewichten zu konstruieren, so dass das Gewicht der Pfade von den zwei Spezies zu ihrem niedrigsten gemeinsamen Vorfahren dem Abstand entspricht. Ein solcher phylogenetischer Baum, der aufgrund von künstlichen evolutionären Distanzen konstruiert wurde, ist in der Abbildung 7.2 illustriert.

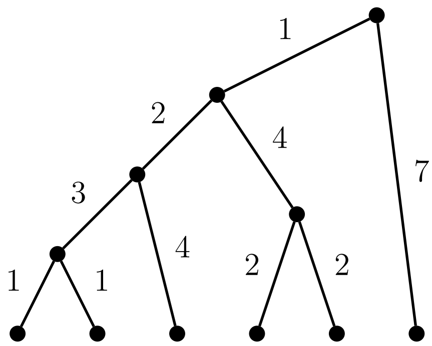


Abbildung 7.2: Beispiel: Ein distanzbasierter phylogenetischer Baum

7.1.2 Charakterbasierte Methoden

Bei den so genannten *charakterbasierten Verfahren* verwendet man gewisse Eigenschaften, so genannte *Charaktere*, der Spezies. Hierbei unterscheidet man *binäre Charaktere*, wie beispielsweise „ist ein Säugetier", „ist ein Wirbeltier", „ist ein Fisch", „ist ein Vogel", „ist ein Lungenatmer", etc., *numerische Charaktere*, wie beispielsweise Anzahl der Extremitäten, Anzahl der Wirbel, etc, und *zeichenreihige Charaktere*, wie beispielsweise bestimmte Teilsequenzen in der DNS. Bei letzterem betrachtet man oft Teilsequenzen aus nicht-codierenden und nicht-regulatorischen Bereichen der DNS, da diese bei Mutationen in der Regel unverändert weitergegeben werden und nicht durch Veränderung einer lebenswichtigen Funktion sofort aussterben.

Das Ziel ist auch hier wieder die Konstruktion eines phylogenetischen Baumes, wobei die Kanten mit Charakteren und ihren Änderungen markiert werden. Eine Markierung einer Kante mit einem Charakter bedeutet hierbei, dass alle Spezies in dem

Teilbaum nun eine Änderung dieses Charakters erfahren. Die genaue Änderung dieses Charakters ist auch an der Kante erfasst.

Bei charakterbasierten Verfahren verfolgt man das Prinzip der minimalen Mutationshäufigkeit bzw. der maximalen Parsimonie (engl. parsimony, Geiz). Das bedeutet, dass man einen Baum sucht, der so wenig Kantenmarkierungen wie möglich besitzt. Man geht hierbei davon aus, dass die Natur keine unnötigen Mutationen verwendet.

In der Abbildung 7.3 ist ein Beispiel für einen solchen charakterbasierten phylogenetischen Baum angegeben, wobei hier nur binäre Charaktere verwendet wurden. Außerdem werden die binären Charaktere hier so verwendet, dass sie nach einer Kantenmarkierung in den Teilbaum eingeführt und nicht gelöscht werden. Bei binären Charakteren kann man dies immer annehmen, da man ansonsten den binären Charakter nur negieren muss.

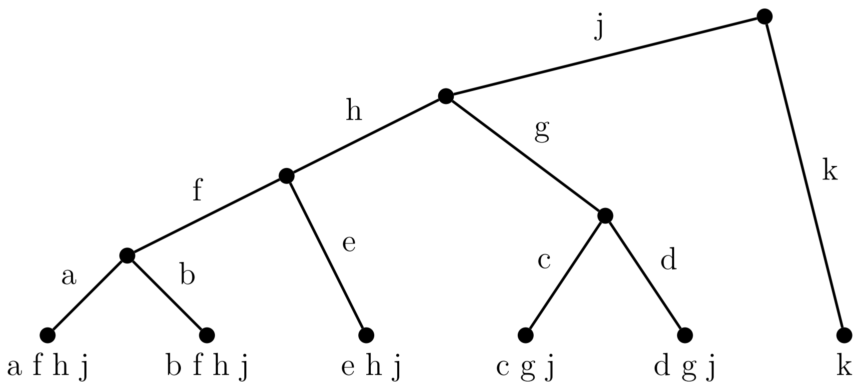


Abbildung 7.3: Beispiel: Ein charakterbasierter phylogenetischer Baum

7.2 Ultrametriken und ultrametrische Bäume

Wir wollen uns zuerst mit distanzbasierten Methoden beschäftigen. Dazu stellen wir zuerst einige schöne und einfache Charakterisierungen vor, ob eine gegebene Distanzmatrix einen phylogenetischen Baum besitzt oder nicht.

7.2.1 Metriken und Ultrametriken

Zuerst müssen wir noch ein paar Eigenschaften von Distanzen wiederholen und einige hier nützliche zusätzliche Definitionen angeben. Zuerst wiederholen wir die Definition einer Metrik (siehe auch Definition 3.4).

Definition 7.2 *Eine Funktion* $d : M^2 \to \mathbb{R}_+$ *heißt* Metrik*, wenn*

(M1) $\forall x, y \in M : d(x, y) = 0 \Leftrightarrow x = y$ *(Definitheit),*

(M2) $\forall x, y \in M : d(x, y) = d(y, x)$ *(Symmetrie),*

(M3) $\forall x, y, z \in M : d(x, z) \leq d(x, y) + d(y, z)$ *(Dreiecksungleichung).*

Im Folgenden werden wir auch die folgende verschärfte Variante der Dreiecksungleichung benötigen.

Definition 7.3 *Eine Metrik heißt* Ultrametrik*, wenn zusätzlich die so genannte* ultrametrische Dreiecksungleichung *gilt:*

$$\forall x, y, z \in M : d(x, z) \leq \max\{d(x, y), d(y, z)\}.$$

Eine andere Charakterisierung der ultrametrischen Ungleichung wird uns im Folgenden aus beweistechnischen Gründen nützlich sein.

Lemma 7.4 *Sei d eine Ultrametrik auf M. Dann sind für alle $x, y, z \in M$ die beiden größten Zahlen aus $d(x, y)$, $d(y, z)$ und $d(x, z)$ gleich.*

Beweis: Zuerst gelte die ultrametrische Dreiecksungleichung für alle $x, y, z \in M$:

$$d(x, z) \leq \max\{d(x, y), d(y, z)\}.$$

Ist $d(x, y) = d(y, z)$, dann ist nichts zu zeigen. Sei also ohne Beschränkung der Allgemeinheit $d(x, y) < d(y, z)$. Dann ist auch $d(x, z) \leq d(y, z)$.

Aufgrund der ultrametrischen Ungleichung gilt ebenfalls:

$$d(y, z) \leq \max\{d(y, x), d(x, z)\} = d(x, z).$$

Die letzte Ungleichung folgt aus der obigen Tatsache, dass $d(y, z) > d(y, x)$.

Zusammen gilt also $d(x, z) \leq d(y, z) \leq d(x, z)$. Also gilt $d(x, z) = d(y, z) > d(x, y)$ und das Lemma ist bewiesen. ■

Nun zeigen wir auch noch die umgekehrte Richtung.

Lemma 7.5 *Sei $d : M^2 \to \mathbb{R}_+$, wobei für alle $x, y \in M$ genau dann $d(x, y) = 0$ gilt, wenn $x = y$. Weiter gelte, dass für alle $x, y, z \in M$ die beiden größten Zahlen aus $d(x, y)$, $d(y, z)$ und $d(x, z)$ gleich sind. Dann ist d eine Ultrametrik.*

Beweis: Die Definitheit (M1) gilt nach Voraussetzung.

Für die Symmetrie (M2) betrachten wir beliebige $x, y \in M$. Aus der Voraussetzung folgt mit $z = x$, dass von $d(x, y)$, $d(y, x)$ und $d(x, x)$ die beiden größten Werte gleich sind. Da nach Voraussetzung $d(x, x) = 0$ sowie $d(x, y) \geq 0$ und $d(y, x) \geq 0$ gilt, folgt, dass $d(x, y) = d(y, x)$ die beiden größten Werte sind und somit nach Voraussetzung gleich sein müssen.

Für die ultrametrische Dreiecksungleichung ist Folgendes zu zeigen:

$$\forall x, y, z \in M : d(x, z) \leq \max\{d(x, y), d(y, z)\}.$$

Wir unterscheiden drei Fälle, je nachdem, welche beiden Werte der drei Distanzen die größten sind und somit nach Voraussetzung gleich sind.

Fall 1 ($d(x, z) \leq d(x, y) = d(y, z)$): Die Behauptung lässt sich sofort verifizieren.

Fall 2 ($d(y, z) \leq d(x, y) = d(x, z)$): Die Behauptung lässt sich sofort verifizieren.

Fall 3 ($d(x, y) \leq d(x, z) = d(y, z)$): Die Behauptung lässt sich sofort verifizieren. ■

Da die beiden Lemmata bewiesen haben, dass von drei Abständen die beiden größten gleich sind, nennt man diese Eigenschaft auch *3-Punkte-Bedingung*.

Zum Schluss dieses Abschnittes definieren wir noch so genannte Distanzmatrizen, aus denen wir im Folgenden die evolutionären Bäumen konstruieren wollen.

Definition 7.6 *Sei $D = (d_{i,j})$ eine symmetrische $n \times n$-Matrix mit $d_{i,i} = 0$ und $d_{i,j} > 0$ für alle $i, j \in [1 : n]$. Dann heißt D eine* Distanzmatrix.

7.2.2 Ultrametrische Bäume

Zunächst einmal definieren wir spezielle evolutionäre Bäume, für die sich, wie wir sehen werden, sehr effizient die gewünschten Bäume konstruieren lassen. Bevor wir diese definieren können, benötigen wir noch den Begriff des niedrigsten gemeinsamen Vorfahren von zwei Knoten in einem Baum.

Definition 7.7 *Sei $T = (V, E)$ ein gewurzelter Baum. Seien $v, w \in V$ zwei Knoten von T. Der* niedrigste gemeinsame Vorfahr *von v und w, bezeichnet als* $\mathrm{lca}(v, w)$ *(engl.* least common ancestor*), ist der Knoten $u \in V$, so dass u sowohl ein Vorfahr von v als auch von w ist und es keinen echten Nachfahren von u gibt, der ebenfalls ein Vorfahr von v und w ist.*

Mit Hilfe des Begriffs des niedrigsten gemeinsamen Vorfahren können wir jetzt ultrametrische Bäume definieren.

Definition 7.8 *Sei D eine $n \times n$-Distanzmatrix. Ein* (strenger) ultrametrischer Baum *T für D ist ein Baum $T = T(D)$ mit*

1. *T besitzt n Blätter, die bijektiv mit $[1 : n]$ markiert sind;*
2. *Jeder innere Knoten von T besitzt mindestens 2 Kinder, die mit Werten aus D markiert sind;*
3. *Entlang eines jeden Pfades von der Wurzel von T zu einem Blatt ist die Folge der Markierungen an den inneren Blättern (streng) monoton fallend;*
4. *Für je zwei Blätter i und j von T ist die Markierung des niedrigsten gemeinsamen Vorfahren gleich d_{ij}.*

In Folgenden werden wir hauptsächlich strenge ultrametrische Bäume betrachten. Wir werden jedoch der Einfachheit wegen immer von ultrametrischen Bäume sprechen. In Abbildung 7.4 ist ein Beispiel für eine 6×6-Matrix angegeben, die einen ultrametrischen Baum besitzt.

Wir wollen an dieser Stelle noch einige Bemerkungen zu ultrametrischen Bäumen festhalten.

- Nicht jede Matrix D besitzt einen ultrametrischen Baum. Dies folgt aus der Tatsache, dass jeder Baum, in dem jeder innere Knoten mindestens zwei Kinder besitzt, maximal $n - 1$ innere Knoten besitzen kann. Dies gilt daher insbesondere für ultrametrische Bäume Also können in Matrizen, die einen ultrametrischen Baum besitzen, nur $n - 1$ von Null verschiedene Werte auftreten.

D	1	2	3	4	5	6
1	0	7	6	7	7	3
2		0	7	4	4	7
3			0	7	7	6
4				0	4	7
5					0	7
6						0

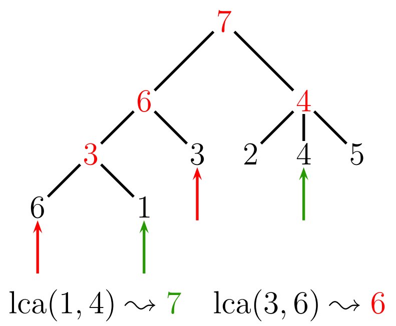


Abbildung 7.4: Beispiel: ultrametrischer Baum

- Der Baum $T(D)$ für D heißt auch *kompakte Darstellung* von D, da sich eine Matrix mit n^2 Einträgen durch einen Baum der Größe $O(n)$ darstellen lässt.
- Die Markierung an den inneren Knoten können als Zeitspanne in die Vergangenheit interpretiert werden. Vor diesem Zeitraum haben sich die Spezies, die in verschiedenen Teilbäumen auftreten, auseinander entwickelt.

Unser Ziel wird es jetzt sein, festzustellen, ob eine gegebene Distanzmatrix einen ultrametrischen Baum besitzt der nicht. Zuerst einmal überlegen wir uns, dass es es sehr viele gewurzelte Bäume mit n Blättern gibt. Somit scheidet ein einfaches Ausprobieren aller möglichen Bäume aus.

Lemma 7.9 *Die Anzahl der ungeordneten binären gewurzelten Bäume mit n Blättern beträgt*

$$\prod_{i=2}^{n}(2i-3) = \frac{(2n-3)!}{2^{n-2}\cdot(n-2)!}.$$

Um ein besseres Gefühl für dieses Anzahl zu bekommen, rechnet man leicht nach, dass

$$\prod_{i=2}^{n}(2i-3) \geq (n-1)! \geq 2^{n-2}$$

gilt. Eine bessere Abschätzung lässt sich natürlich mit Hilfe der Stirlingschen Formel bekommen.

Beweis: Wir führen den Beweis durch vollständige Induktion über n.

Induktionsanfang ($n = 2$): Hierfür gilt die Formel offensichtlich, das es genau einem Baum mit zwei markierten Blättern gibt.

Induktionsanfang ($n-1 \to n$): Sei T ein ungeordneter binärer gewurzelter Baum mit n Blättern. Der Einfachheit nehmen wir im Folgenden an, dass unser Baum noch eine Superwurzel besitzt, deren einziges Kind die ursprüngliche Wurzel von T ist.

Wir entfernen jetzt das Blatt v mit der Markierung n. Der Elter w davon hat jetzt nur noch ein Kind und wir entfernen es ebenfalls. Dazu wird das andere Kind von w jetzt ein Kind des Elters von w (anstatt von w). Den so konstruierten Baum nennen wir T'. Wir merken noch an, dass genau eine Kante eine „Erinnerung“ an das Entfernen von v und w hat. Falls w die Wurzel war, so bleibt die Superwurzel im Baum und die Kante von der Superwurzel hat sich das Entfernen „gemerkt“.

Wir stellen fest, dass T' ein ungeordneter binärer gewurzelter Baum ist. Davon gibt es nach Induktionsvoraussetzung $\prod_{i=2}^{n-1}(2i-3)$ viele. Darin kann an jeder Kante v mit seinem Elter w entfernt worden sein.

Wie viele Kanten besitzt ein binärer gewurzelter Baum mit $n-1$ Blättern? Ein binärer gewurzelter Baum mit $n-1$ Blättern besitzt genau $n-2$ innere Knoten plus die von uns hinzugedachte Superwurzel. Somit besitzt der Baum

$$(n-1)+(n-2)+1 = 2n-2$$

Knoten. Da in einem Baum die Anzahl der Kanten um eines niedriger ist als die Anzahl der Knoten, besitzt unser Baum $2n-3$ Kanten, die sich an einen Verlust eines Blattes „erinnern“ können. Somit ist die Gesamtanzahl der ungeordneten binären gewurzelten Bäume mit n Blättern genau

$$(2n-3)\cdot\prod_{i=2}^{n-1}(2i-3) = \prod_{i=2}^{n}(2i-3)$$

und der Induktionschluss ist vollzogen. ■

Wir fügen noch eine ähnliche Behauptung für die Anzahl ungewurzelter Bäume an. Der Beweis ist im Wesentlichen ähnlich zu dem vorherigen.

Lemma 7.10 *Die Anzahl der ungewurzelten (freien) Bäume mit n Blättern, deren innere Knoten jeweils den Grad 3 besitzen, beträgt*

$$\prod_{i=3}^{n}(2i-5) = \frac{(2n-5)!}{2^{n-3}\cdot(n-3)!}.$$

Wir benötigen jetzt noch eine kurze Definition, die Distanzmatrizen und Metriken in Beziehung setzen.

Definition 7.11 *Eine $n \times n$-Distanzmatrix M* induziert *eine Metrik bzw. Ultrametrik auf $[1:n]$, wenn die Funktion $d : [1:n]^2 \to \mathbb{R}_+$ mit $d(x,y) = M_{x,y}$ eine Metrik bzw. Ultrametrik ist.*

Mit Hilfe oben erwähnten Charakterisierung einer Ultrametrik, dass von den drei Abständen zwischen drei Punkten, die beiden größten gleich sind, können wir sofort einen einfachen Algorithmus zur Erkennung ultrametrischer Matrizen angeben. Wir müssen dazu nur alle dreielementigen Teilmengen aus $[1:n]$ untersuchen, ob von den drei verschiedenen Abständen, die beiden größten gleich sind. Falls ja, ist die Matrix ultrametrisch, ansonsten nicht.

Dieser Algorithmus hat jedoch eine Laufzeit $O(n^3)$. Wir wollen im Folgenden einen effizienteren Algorithmus zur Konstruktion ultrametrischer Matrizen angeben, der auch gleichzeitig noch die zugehörigen ultrametrischen Bäume mitberechnet.

7.2.3 Charakterisierung ultrametrischer Bäume

Wir geben jetzt eine weitere Charakterisierung ultrametrischer Matrizen an, deren Beweis sogar einen effizienten Algorithmus zur Konstruktion ultrametrischer Bäume erlaubt.

Theorem 7.12 *Eine symmetrische $n \times n$-Distanzmatrix D besitzt genau dann einen ultrametrischen Baum, wenn D eine Ultrametrik induziert.*

Beweis: **$\Rightarrow$:** Die Definitheit und Symmetrie folgt unmittelbar aus der Definition einer Distanzmatrix. Wir müssen also nur noch die ultrametrische Dreiecksungleichung zeigen:

$$\forall i,j,k \in [1:n] : \ d_{ij} \leq \max\{d_{ik}, d_{jk}\}.$$

Wir betrachten dazu den ultrametrischen Baum $T = T(D)$. Wir unterscheiden dazu drei Fälle in Abhängigkeit, wie sich die niedrigsten gemeinsamen Vorfahren von i, j und k zueinander verhalten. Sei dazu $x = \mathrm{lca}(i,j)$, $y = \mathrm{lca}(i,k)$ und $z = \mathrm{lca}(j,k)$. In einem Baum muss dabei gelten, dass mindestens zwei der drei Knoten identisch sind. Die ersten beiden Fälle sind in Abbildung 7.5 dargestellt.

Fall 1 ($\boldsymbol{y = z \neq x}$): Damit folgt aus dem rechten Teil der Abbildung 7.5 sofort, dass $d(i,j) \leq d(i,k) = d(j,k)$.

Fall 2 ($\boldsymbol{x = y \neq z}$): Damit folgt aus dem linken Teil der Abbildung 7.5 sofort, dass $d(j,k) \leq d(i,k) = d(i,j)$.

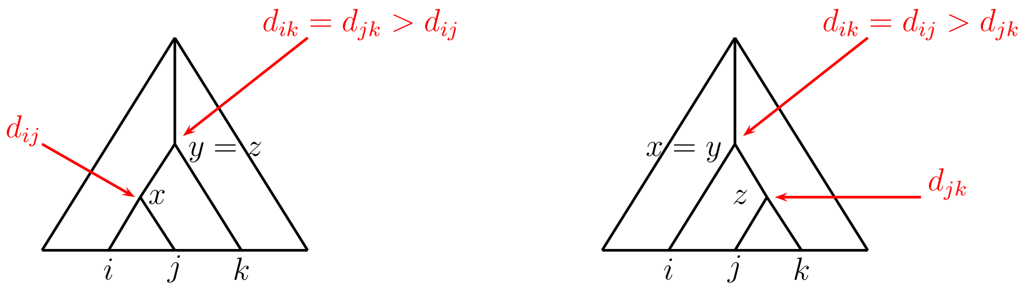


Abbildung 7.5: Skizze: Fall 1 und Fall 2

Fall 3 ($x = z \neq y$): Dieser Fall ist symmetrisch zu Fall 2 (einfaches Vertauschen von i und j).

Fall 4 ($x = y = z$): Aus der Abbildung 7.6 folgt auch hier, dass die ultrametrische Dreiecksungleichung gilt, da alle drei Abstände gleich sind. Wenn man statt

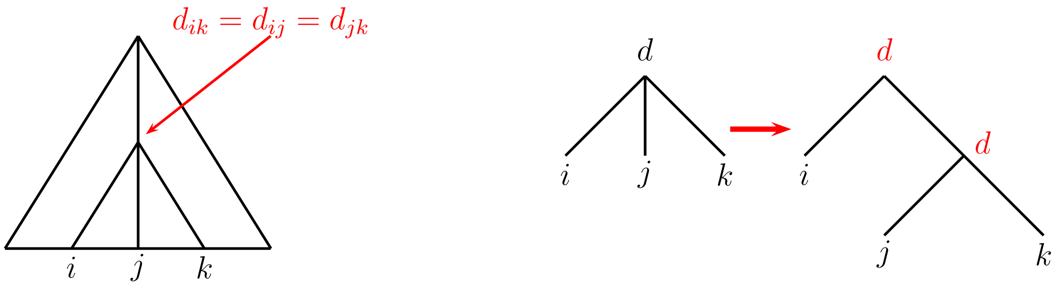


Abbildung 7.6: Skizze: Fall 4 und binäre Neukonstruktion

eines strengen ultrametrischen Baumes lieber einen binären ultrametrischen Baum haben möchte, so kann man ihn beispielsweise wie im rechten Teil der Abbildung 7.6 umbauen.

$\Leftarrow$: Wir betrachten zuerst die Abstände von Blatt 1 zu allen anderen Blättern. Sei also $\{d_{11}, \ldots, d_{1n}\} = \{\delta_1, \ldots, \delta_k\}$, d.h. $\delta_1, \ldots, \delta_k$ sind die paarweise verschiedenen Abstände, die vom Blatt 1 aus auftreten. Ohne Beschränkung der Allgemeinheit nehmen wir dabei an, dass $\delta_1 < \cdots < \delta_k$. Wir partitionieren dann $[2 : n]$ wie folgt:

$$D_i = \{\ell \in [2 : n] \ : \ d_{1\ell} = \delta_i\}.$$

Es gilt dann offensichtlich $[2 : n] = \biguplus_{i=1}^{k} D_i$. Wir bestimmen jetzt für die Mengen D_i rekursiv die entsprechenden ultrametrischen Bäume. Anschließend konstruieren wir einen Pfad von der Wurzel zum Blatt 1 mit k inneren Knoten, an die die rekursiv konstruierten Teilbäume $T(D_i)$ angehängt werden. Dies ist in Abbildung 7.7 schematisch dargestellt.

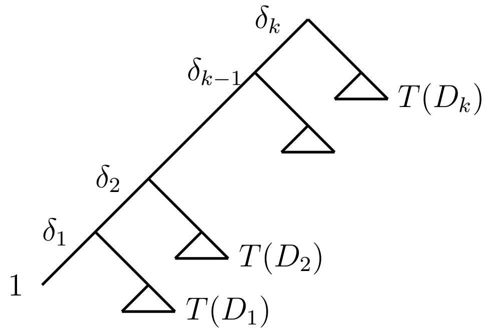


Abbildung 7.7: Skizze: Rekursive Konstruktion ultrametrischer Bäume

Wir müssen jetzt nachprüfen, ob der konstruierte Baum ultrametrisch ist. Dazu müssen wir zeigen, dass die Knotenmarkierungen auf einem Pfad von der Wurzel zu einem Blatt streng monoton fallend sind und dass der Abstand von den Blättern i und j gerade die Markierung von $\mathrm{lca}(i, j)$ ist.

Für den ersten Teil überlegen wir uns, dass diese sowohl auf dem Pfad von der Wurzel zu Blatt 1 gilt als auch in den Teilbäumen $T(D_i)$. Wir müssen nur noch die Verbindungspunkte überprüfen. Dies ist in Abbildung 7.8 illustriert. Hier sind x und

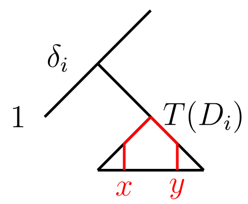


Abbildung 7.8: Skizze: Abstände von x und y und geforderte Monotonie

y zwei Blättern in $T(D_i)$, deren niedrigster gemeinsamer Vorfahre die Wurzel von $T(D_i)$ ist. Wir müssen als zeigen, dass $d_{x,y} < \delta_i$ gilt. Es gilt zunächst aufgrund der ultrametrischen Dreiecksungleichung:

$$d_{xy} \leq \max\{d_{1x}, d_{1y}\} = d_{1x} = d_{1y} = \delta_i.$$

Gilt jetzt $d_{xy} < \delta_i$ dann ist alles gezeigt. Andernfalls gilt $\delta_{xy} = \delta_i$ und wir werden den Baum noch ein wenig umbauen, wie in der folgenden Abbildung 7.9 illustriert. Dabei wird die Wurzel des Teilbaums $T(D_i)$ mit dem korrespondierenden Knoten des Pfades von der Wurzel des Gesamtbaumes zum Blatt 1 miteinander identifiziert und die Kante dazwischen gelöscht. Damit haben wir die strenge Monotonie der Knotenmarkierungen auf den Pfaden von der Wurzel zu den Blättern nachgewiesen.

Es ist jetzt noch zu zeigen, dass die Abstände von zwei Blättern x und y den Knotenmarkierungen entsprechen. Innerhalb der Teilbäume $T(D_i)$ gilt dies nach Kon-

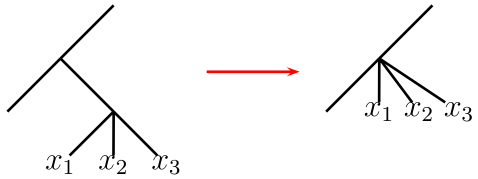


Abbildung 7.9: Skizze: Umbau im Falle nicht-strenger Monotonie

struktion. Ebenfalls gilt dies nach Konstruktion für das Blatt 1 mit allen anderen Blättern.

Wir müssen diese Eigenschaft nur noch nachweisen, wenn sich zwei Blätter in unterschiedlichen Teilbäumen befinden. Sei dazu $x \in V(T(D_i))$ und $y \in V(T(D_j))$, wobei wir ohne Beschränkung der Allgemeinheit annehmen, dass $\delta_i > \delta_j$ gilt. Dies ist in der Abbildung 7.10 illustriert.

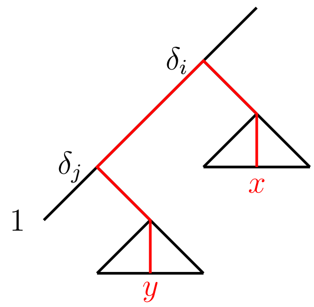


Abbildung 7.10: Skizze: Korrektheit der Abstände zweier Blätter in unterschiedlichen rekursiv konstruierten Teilbäumen

Nach Konstruktion gilt: $\delta_i = d_{1x}$ und $\delta_j = d_{1y}$. Mit Hilfe der ultrametrischen Dreiecksungleichung folgt:

$$\begin{aligned} d_{xy} &\leq \max\{d_{1x}, d_{1y}\} \\ &= \max\{\delta_i, \delta_j\} \\ &= \delta_i \\ &= d_{1x} \\ &\leq \max\{d_{1y}, d_{yx}\} \\ & \text{da } d_{1y} = \delta_j < \delta_i = d_{1x} \\ &= d_{xy} \end{aligned}$$

Daraus folgt also $d_{xy} \leq \delta_i \leq d_{xy}$ und somit $\delta_i = d_{xy}$. Damit ist der Beweis abgeschlossen. ■

Aus dem Beweis folgt unter Annahme der strengen Monotonie (d.h. für strenge ultrametrische Bäume), dass der konstruierte ultrametrische Baum eindeutig ist. Die Konstruktion des ultrametrischen Baumes ist ja bis auf Umordnung der Kinder eines Knoten eindeutig festgelegt.

Korollar 7.13 *Sei D eine streng ultrametrische Matrix, dann ist der zugehörige strenge ultrametrische Baum eindeutig.*

Für nicht-strenge ultrametrische Bäume kann man sich überlegen, wie die Knoten mit gleicher Markierung umgeordnet werden können, so dass der Baum ein ultrametrischer bleibt. Bis auf diese kleinen Umordnungen sind auch nicht-streng ultrametrische Bäume im Wesentlichen eindeutig.

7.2.4 Konstruktion ultrametrischer Bäume

Wir versuchen jetzt aus dem Beweis der Existenz eines ultrametrischen Baumes einen effizienten Algorithmus zur Konstruktion eines ultrametrischen Baumes zu entwerfen und dessen Laufzeit zu analysieren.

Wir erinnern noch einmal an die Partition von $[2 : n]$ durch $D_i = \{\ell : d_{1\ell} = \delta_i\}$ mit $n_i := |D_i|$. Dabei war $\{d_{11}, \ldots, d_{1n}\} = \{\delta_1, \ldots, \delta_k\}$, wobei $\delta_1 < \cdots < \delta_k$. Wir erinnern hier auch noch einmal an Skizze der Konstruktion des ultrametrischen Baumes, wie in Abbildung 7.11.

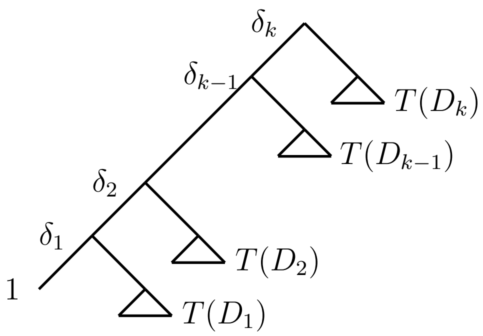


Abbildung 7.11: Skizze: Konstruktion eines ultrametrischen Baumes

Daraus ergibt sich der erste naive Algorithmus, der in Abbildung 7.12 aufgelistet ist. Da jeder Schritt Zeitbedarf $O(n \log(n))$ und es maximal n rekursive Aufrufe geben kann (mit jedem rekursiven Aufruf wird ein Knoten des ultrametrischen Baumes explizit konstruiert), ist die Laufzeit insgesamt $O(n^2 \log(n))$.

1. Sortiere die Menge $\{d_{11}, \ldots, d_{1n}\}$ und bestimme anschließend $\delta_1, \ldots, \delta_k$ mit $\{\delta_1, \ldots, \delta_k\} = \{d_{11}, \ldots, d_{1n}\}$ und partitioniere $[2:n] = \biguplus_{i=1}^{k} D_i$. $O(n\log(n))$
2. Bestimme für $D_1, \ldots D_k$ ultrametrische Bäume $T(D_1), \ldots, T(D_k)$ mittels Rekursion. $\sum_{i=1}^{k} T(n_j)$
3. Setze die Teillösungen und den Pfad von der Wurzel zu Blatt 1 zur Gesamtlösung zusammen. $O(k) = O(n)$

Abbildung 7.12: Algorithmus: Naive Konstruktion eines ultrametrischen Baumes

Wir werden jetzt noch einen leicht modifizierten Algorithmus vorstellen, der eine Laufzeit von nur $O(n^2)$ besitzt. Dies ist optimal, da die Eingabe, die gegebene Distanzmatrix, bereits eine Größe von $\Theta(n^2)$ besitzt.

Dazu beachten wir, dass der aufwendige Teil des Algorithmus das Sortieren der Elemente in $\{d_{11}, \ldots, d_{1n}\}$ ist. Insbesondere ist dies sehr teuer, wenn es nur wenige verschieden Elemente in dieser Menge gibt, da dann auch nur entsprechend wenig rekursive Aufrufe folgen.

Daher werden wir zuerst feststellen, wie viele verschiedene Elemente es in der Menge $\{d_{11}, \ldots, d_{1n}\}$ gibt und bestimmen diese. Die paarweise verschiedenen Elemente dieser Menge können wir in einer linearen Liste (oder auch in einem balancierten Suchbaum) aufsammeln. Dies lässt sich in Zeit $O(k \cdot n)$ (bzw. in Zeit $O(n\log(k))$ bei Verwendung balancierter Bäume) implementieren. Anschließend müssen wir nur noch k Elemente sortieren. Der Algorithmus selbst ist in Abbildung 7.13 aufgelistet.

1. Bestimme zuerst $k = |\{d_{11}, \ldots, d_{1n}\}|$ und $\{\delta_1, \ldots, \delta_k\} = \{d_{11}, \ldots, d_{1n}\}$.

   Dies kann mit Hilfe linearer Listen in Zeit $O(k \cdot n)$ erledigt werden. Mit Hilfe balancierter Bäume können wir dies sogar in Zeit $O(n\log(k))$ realisieren.
2. Sortiere die k paarweise verschiedenen Werte $\{\delta_1, \ldots, \delta_k\}$.

   Dies kann in Zeit $O(k\log(k)$ erledigt werden.
3. Bestimme die einzelnen Teilbäume $T(D_i)$ rekursiv.
4. Setze die Teillösungen und den Pfad von der Wurzel zu Blatt 1 zur Gesamtlösung zusammen.

   Dies lässt sich wiederum in Zeit $O(k)$ realisieren

Abbildung 7.13: Algorithmus: Konstruktion eines ultrametrischen Baumes

Damit erhalten wir als Rekursionsgleichung für diesen modifizierten Algorithmus:

$$T(n) = d \cdot k \cdot n + \sum_{i=1}^{k} T(n_i)$$

mit $\sum_{i=1}^{k} n_i = n - 1$, wobei $n_i \geq 1$ für $i \in [1 : n]$, und einer geeignet gewählten Konstanten d. Hierbei ist zu beachten, dass $O(k \log(k)) = O(k \cdot n)$ ist, da ja $k < n$ ist.

Lemma 7.14 *Ist D eine ultrametrische $n \times n$-Matrix, dann kann der zugehörige ultrametrische Baum in Zeit $O(n^2)$ konstruiert werden.*

Beweis: Es ist nur noch zu zeigen, dass $T(n) \leq c \cdot n^2$ für eine geeignet gewählte Konstante c gilt. Wir wählen c jetzt so, dass zum einen $c \geq 2d$ und zum anderen $T(n) \leq c \cdot n^2$ für alle $n \leq 3$ gilt. Den Beweis selbst führen wir mit vollständiger Induktion über n.

Induktionsanfang ($n \leq 3$): Nach Wahl von c gilt dies offensichtlich.

Induktionsschritt ($\rightarrow n$): Es gilt dann nach Konstruktion des Algorithmus mit $n = \sum_{i=1}^{k} n_i$:

$$\begin{aligned}
T(n) &\leq d \cdot k \cdot n + \sum_{i=1}^{k} T(n_i) \\
&\qquad \text{nach Induktionsvoraussetzung ist } T(n_i) \leq c \cdot n_i^2 \\
&\leq d \cdot k \cdot n + \sum_{i=1}^{k} c \cdot n_i^2 \\
&= d \cdot k \cdot n + c \sum_{i=1}^{k} n_i(n - n + n_i) \\
&= d \cdot k \cdot n + c \sum_{i=1}^{k} n_i \cdot n - c \sum_{i=1}^{k} n_i(n - n_i) \\
&\qquad \text{da } x(c - x) > 1(c-1) \text{ für } x \in [1 : c-1] \text{, siehe auch Abbildung 7.14} \\
&\leq d \cdot k \cdot n + cn^2 - c \sum_{i=1}^{k} 1(n-1)
\end{aligned}$$

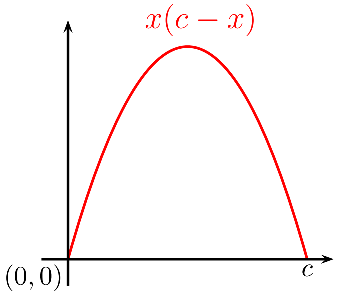


Abbildung 7.14: Skizze: Die Funktion $[0, c] \to \mathbb{R}_+ : x \mapsto x(c - x)$

$$
\begin{aligned}
&\leq d \cdot k \cdot n + c \cdot n^2 - c \cdot k(n-1) \\
&\quad \text{da } c \geq 2d \\
&\leq d \cdot k \cdot n + c \cdot n^2 - 2d \cdot k(n-2) \\
&\quad \text{da } 2(n-1) \geq n \text{ für } n \geq 3 \\
&\leq d \cdot k \cdot n + c \cdot n^2 - d \cdot k \cdot n \\
&= c \cdot n^2.
\end{aligned}
$$

Damit ist der Induktionsschluss vollzogen und der Beweis beendet. ■

7.3 Additive Distanzen und Bäume

Leider sind nicht alle Distanzmatrizen ultrametrisch. Wir wollen jetzt ein größere Klasse von Matrizen vorstellen, zu denen sich evolutionäre Bäume konstruieren lassen, sofern wir auf eine explizite Wurzel in diesen Bäumen verzichten können.

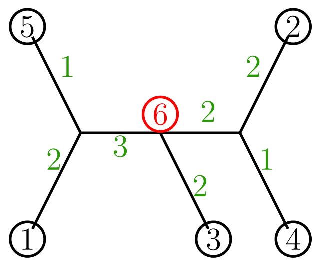
7.3.1 Additive Bäume

Zunächst einmal definieren wir, was wir unter additiven Bäumen, d.h. evolutionären Bäumen ohne Wurzel, verstehen wollen.

Definition 7.15 *Sei D eine $n \times n$-Distanzmatrix. Sei T ein Baum mit mindestens n Knoten und positiven Kantengewichten, wobei einige Knoten bijektiv mit Werten aus $[1 : n]$ markiert sind. Dann ist T ein* additiver Baum *für D, wenn der Pfad vom Knoten mit Markierung i zum Knoten mit Markierung j in T das Gewicht d_{ij} besitzt.*

Wie bei den ultrametrischen Bäumen besitzt auch nicht jede Distanzmatrix einen additiven Baum. Des Weiteren sind nicht markierte Blätter in einem additiven Baum überflüssig, da jeder Pfad innerhalb eines Baum nur dann ein Blatt berührt, wenn dieses ein Endpunkt des Pfades ist. Daher nehmen wir im Folgenden ohne Beschränkung der Allgemeinheit an, dass ein additiver Baum nur markierte Blätter besitzt. In der folgenden Abbildung 7.15 ist noch einmal ein Beispiel einer Matrix samt ihres zugehörigen additiven Baumes angegeben.

D	1	2	3	4	5	6
1	0	9	7	8	3	5
2		0	6	3	8	4
3			0	5	6	2
4				0	7	3
5					0	4
6						0



Gewicht des Pfades zwischen 5 und 4: $\Rightarrow 1 + 3 + 2 + 1 = 7$

Gewicht des Pfades zwischen 1 und 5: $\Rightarrow 2 + 1 = 3$

Abbildung 7.15: Beispiel: Eine Matrix und der zugehörige additive Baum

Definition 7.16 *Sei D eine Distanzmatrix. Besitzt D einen additiven Baum, so heißt D eine* additive Matrix. *Ein additiver Baum heißt* kompakt, *wenn alle Knoten markiert sind (insbesondere auch die inneren Knoten). Ein additiver Baum heißt* extern, *wenn nur Blätter markiert sind.*

In der Abbildung 7.15 ist der Baum ohne Knoten 6 (und damit die Matrix ohne Zeile und Spalte 6) ein externer additiver Baum. Durch Hinzufügen von zwei Markierungen (und zwei entsprechenden Spalten und Zeilen in der Matrix) könnte dieser Baum zu einem kompakten additiven Baum gemacht werden.

Lemma 7.17 *Sei D eine additive Matrix, dann induziert D eine Metrik.*

Beweis: Da D eine Distanzmatrix ist, gelten die Definitheit und Symmetrie unmittelbar. Die Dreiecksungleichung sieht man leicht, wenn man sich die entsprechenden Pfade im zu D gehörigen additiven Baum betrachtet. ■

Die Umkehrung gilt übrigens nicht, wie wir später noch zeigen werden.

7.3.2 Charakterisierung additiver Bäume

Wir wollen nun eine Charakterisierung additiver Matrizen angeben, mit deren Hilfe sich auch gleich ein zugehöriger additiver Baum konstruieren lässt. Zunächst einmal zeigen wir, dass ultrametrische Bäume spezielle additive Bäume sind.

Lemma 7.18 *Sei D eine additive Matrix. D ist genau dann ultrametrisch, wenn es einen additiven Baum T für D gibt, so dass es in T einen Knoten v (zentraler Knoten) gibt, der zu allen markierten Knoten denselben Abstand besitzt.*

Beweis: **⇒:** Sei T ein ultrametrischer Baum für D mit Knotenmarkierungen μ. Wir erhalten daraus einen additiven Baum T', indem wir als Kantengewicht einer Kante (v, w) folgendes wählen:

$$\gamma(v, w) = \frac{1}{2}(\mu(v) - \mu(w)).$$

Man rechnet jetzt leicht nach, dass für zwei Blätter i und j das sowohl das Gewicht des Weges von i nach $\mathrm{lca}(i, j)$ als auch das Gewicht von j nach $\mathrm{lca}(i, j)$ gerade $\frac{1}{2} \cdot d_{ij}$ beträgt. Die Länge des Weges von i nach j beträgt daher im additiven Baum wie gefordert d_{ij}.

Falls einen Knoten mit Grad 2 in einem solchen additiven Baum stören (wie sie beispielweise von der Wurzel konstruiert werden), der kann diese, wie in Abbildung 7.16 illustriert, eliminieren. In dieser Abbildung stellt der rote Knoten den zentralen Knoten des additiven Baumes dar.

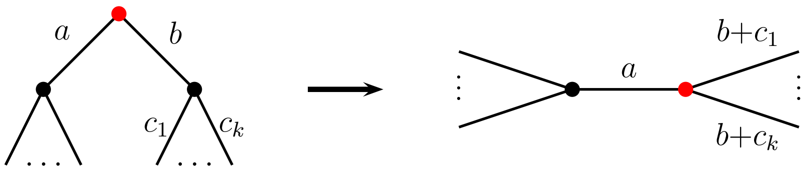


Abbildung 7.16: Skizze: Elimination von Knoten mit Grad 2

⇐: Sei D additiv und sei v der Knoten des additiven Baums T, der von allen Blättern den gleichen Abstand hat. Man überlegt sich leicht, dass T dann extern additiv sein muss.

Betrachtet man v als Wurzel, so gilt für beliebige Blätter i, j, dass der Abstand zwischen dem kleinsten gemeinsamen Vorfahren ℓ für beide Blätter gleich ist. (Da der Abstand $d_{\ell v}$ fest ist, wäre andernfalls der Abstand der Blätter zu v nicht gleich). Wir zeigen jetzt, dass für drei beliebige Blätter i, j, k die ultrametrische Dreiecksungleichung $d_{ik} \leq \max\{d_{ij}, d_{jk}\}$ gilt. Sei dazu $\mathrm{lca}(i, j)$ der kleinste gemeinsame Vorfahre von i, j.

Falls $\mathrm{lca}(i, j) = \mathrm{lca}(i, k) = \mathrm{lca}(j, k)$ die gleichen Knoten sind, so ist $d_{ik} = d_{ij} = d_{jk}$ und die Dreiecksungleichung gilt mit Gleichheit.

Daher sei jetzt ohne Beschränkung der Allgemeinheit $\mathrm{lca}(i, j) \neq \mathrm{lca}(i, k)$ und dass $\mathrm{lca}(i, k)$ näher an der Wurzel liegt. Da $\mathrm{lca}(i, j)$ und $\mathrm{lca}(i, k)$ auf dem Weg von i zur Wurzel liegen gilt entweder $\mathrm{lca}(j, k) = \mathrm{lca}(j, i)$ oder $\mathrm{lca}(j, k) = \mathrm{lca}(i, k)$ sein.

Sei ohne Beschränkung der Allgemeinheit $\mathrm{lca}(j, k) = \mathrm{lca}(i, k) = \mathrm{lca}(i, j, k)$. Dann gilt:

$$\begin{aligned}
d_{ij} &= w(i, \mathrm{lca}(i, j)) + w(j, \mathrm{lca}(i, j)) \\
&= 2 \cdot w(j, \mathrm{lca}(i, j)), \\[1ex]
d_{ik} &= w(i, \mathrm{lca}(i, j)) + w(\mathrm{lca}(i, j), \mathrm{lca}(i, j, k)) + w(\mathrm{lca}(i, j, k), k) \\
&= 2 \cdot w(\mathrm{lca}(i, j, k), k), \\[1ex]
d_{jk} &= w(j, \mathrm{lca}(i, j)) + w(\mathrm{lca}(i, j), \mathrm{lca}(i, j, k)) + w(\mathrm{lca}(i, j, k), k) \\
&= 2 \cdot w(\mathrm{lca}(i, j, k), k)
\end{aligned}$$

Daher gilt unter anderem $d_{ij} \leq d_{ik}$, $d_{ik} \leq d_{jk}$ und $d_{jk} \leq d_{ik}$. Somit ist die Dreiecksungleichung immer erfüllt. ∎

Wie können wir nun für eine Distanzmatrix entscheiden, ob sie additiv ist, und falls ja, beschreiben, wie der zugehörige additive Baum aussieht? Im vorherigen Lemma haben wir gesehen, wie wir das Problem auf ultrametrische Matrizen zurückführen können.

Wir wollen dies noch einmal mir einer anderen Charakterisierung tun. Wir betrachten zuerst die gegebene Matrix D. Diese ist genau dann additiv, wenn sie einen additiven Baum T_D besitzt. Wenn wir diesen Baum wurzeln und die Kanten zu den Blättern so verlängernd, dass alle Pfade von der Wurzel zu den Blättern gleiche Gewicht besitzen, dann ist T'_D ultrametrisch. Daraus können wir dann eine Distanzmatrix $D(T'_D)$ ablesen, die ultrametrisch ist, wenn D additiv ist. Dies ist in der folgenden Abbildung 7.17 schematisch dargestellt.

Wir wollen also die gebebene Matrix D so modifizieren, dass daraus eine ultrametrische Matrix wird. Wenn diese Idee funktioniert können wir eine Matrix auf Additivität hin testen, indem wir die zugehörige, neu konstruierte Matrix auf Ultrametrik hin testen. Für Letzteres haben wir ja bereits einen effizienten Algorithmus kennen gelernt.

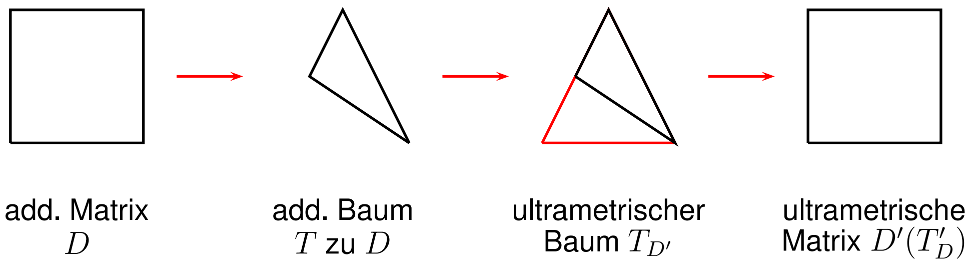


Abbildung 7.17: Skizze:

Sei im Folgenden D eine additive Matrix und d_{ij} ein maximaler Eintrag,d h.

$$d_{ij} \geq \max\{d_{k\ell} \;:\; k, \ell \in [1:n]\}\,.$$

Weiter sei T_D der additive Baum zu D. Wir wurzeln jetzt diesen Baum am Knoten i. Man beachte, dass i ein Blatt ist. Wir kommen später noch darauf zurück, wie wir dafür sorgen dass i ein Blatt bleibt Dies ist in der folgenden Abbildung 7.18 illustriert.

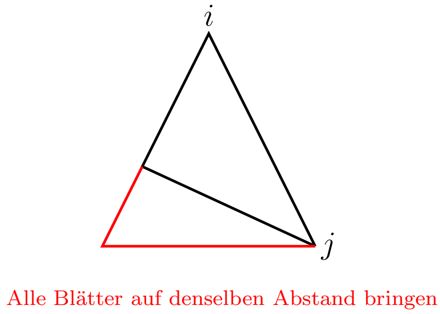


Abbildung 7.18: Skizze: Wurzeln von T_D am Blatt i

Unser nächster Schritt besteht jetzt darin, alle Blätter (außer i) auf denselben Abstand zur neuen Wurzel zu bringen. Wir betrachten dazu jetzt ein Blatt k des neuen gewurzelten Baumes. Wir versuchen die zu k inzidente Kante jetzt so zu verlängern, dass der Abstand von k zur Wurzel i auf d_{ij} anwächst. Dazu setzen wir das Kantengewicht auf d_{ij} und verkürzen es um das Gewicht des restlichen Pfades von k zu i, d.h. um $(d_{ik} - d)$, wobei d das Gewicht der zu k inzidenten Kante ist. Dies ist in Abbildung 7.19 noch einmal illustriert. War der Baum vorher additiv, so ist er jetzt ultrametrisch, wenn wir als Knotenmarkierung jetzt das Gewicht des Pfades einen Knotens zu einem seiner Blätter (die jetzt alle gleich sein müssen) wählen.

Somit haben wir aus einem additiven einen ultrametrischen Baum gemacht. Jedoch haben wir dazu den additiven Baum benötigt, den wir eigentlich erst konstruieren

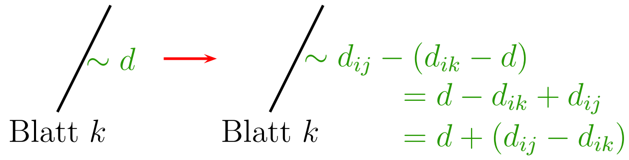


Abbildung 7.19: Skizze: Verlängern der zu Blättern inzidenten Kanten

wollen. Es stellt sich nun die Frage, ob wir die entsprechende ultrametrische Matrix aus der additiven Matrix direkt, ohne Kenntnis des zugehörigen additiven Baumes berechnen können. Betrachten wir dazu noch einmal zwei Blätter im additiven Baum und versuchen den entsprechenden Abstand im ultrametrischen Baum berechnen. Dazu betrachten wir die Abbildung 7.20.

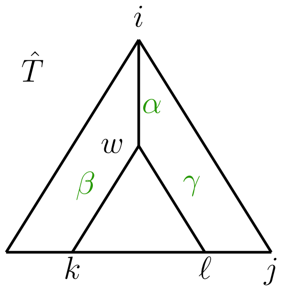


Abbildung 7.20: Skizze: Abstände im gewurzelten additiven Baum $\hat{T}$

Seien k und ℓ zwei Blätter, für die wir den Abstand in der entsprechenden ultrametrischen Matrix bestimmen wollen. Sei w der niedrigste gemeinsame Vorfahre von k und ℓ im gewurzelten additiven Baum $\hat{T}$ und α, β bzw. γ die Abstände im gewurzelten additiven Baum $\hat{T}$ zwischen der Wurzel und w, w und k bzw. w und ℓ. Es gilt dann

$$\begin{aligned} \beta &= d_{ik} - \alpha \\ \gamma &= d_{il} - \alpha \end{aligned}$$

Im ultrametrischen Baum T'_D gilt dann:

$$\begin{aligned} d_{T'}(w,k) &= d_{T'}(w,\ell) \\ &= \beta + (d_{ij} - d_{ik}) \\ &= \gamma + (d_{ij} - d_{i\ell}). \end{aligned}$$

Damit gilt:

$$d_{T'}(w,k) = \beta + d_{ij} - d_{ik}$$

$$\begin{aligned} &= d_{ik} - \alpha + d_{ij} - d_{ik} \\ &= d_{ij} - \alpha \end{aligned}$$

und analog:

$$\begin{aligned} d_{T'}(w, \ell) &= \gamma + d_{ij} - d_{i\ell} \\ &= d_{i\ell} - \alpha + d_{ij} - d_{i\ell} \\ &= d_{ij} - \alpha. \end{aligned}$$

Wir müssen jetzt nur noch α bestimmen. Aus der Skizze in Abbildung 7.20 folgt sofort, wenn wir die Gewichte der Pfade von i nach k sowie ℓ addieren und davon das Gewicht des Pfades von k nach ℓ subtrahieren:

$$2\alpha = d_{ik} + d_{i\ell} - d_{k\ell}.$$

Daraus ergibt sich:

$$\begin{aligned} d_{T'}(w, k) &= d_{ij} - \frac{1}{2}(d_{ik} + d_{i\ell} - d_{k\ell}), \\ d_{T'}(w, \ell) &= d_{ij} - \frac{1}{2}(d_{ik} + d_{i\ell} - d_{k\ell}). \end{aligned}$$

Somit können wir jetzt die ultrametrische Matrix D' direkt aus der additiven Matrix D berechnen:

$$d'_{k\ell} := d_{T'}(w, k) = d_{T'}(w, \ell) = d_{ij} - \frac{1}{2}(d_{i\ell} + d_{ik} - d_{k\ell}).$$

Wir müssen uns jetzt nur noch überlegen, dass dies wirklich eine ultrametrische Matrix ist, da wir den additiven Baum ja am Blatt i gewurzelt haben. Damit würden wir sowohl ein Blatt verlieren, nämlich i, als auch keinen echten ultrametrischen Baum generieren, da dessen Wurzel nur ein Kind anstatt mindestens zweier besitzt. In Wirklichkeit wurzeln wir den additiven Baum am zu i adjazenten Knoten, wie in Abbildung 7.21 illustriert. Damit erhalten wir einen echten ultrametrischen Baum.

Damit haben wir das folgende Lemma bewiesen.

Lemma 7.19 *Sei D eine additive Matrix, deren maximaler Eintrag d_{ij} ist, dann ist D' mit*

$$d'_{kl} = d_{ij} - \frac{1}{2}(d_{i\ell} + d_{ik} - d_{kl\ell})$$

eine ultrametrische Matrix.

Es wäre schön, wenn auch die umgekehrte Richtung gelten würde, nämlich, dass wenn D' ultrametrisch ist, dass dann bereits D additiv ist. Dies ist leider nicht der

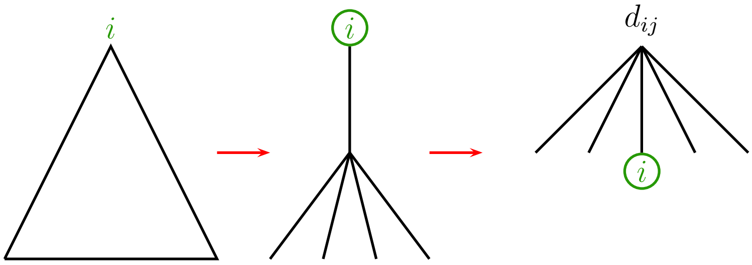


Abbildung 7.21: Skizze: Wirkliches Wurzeln des additiven Baumes

D	1	2	3
1	0	8	4
2		0	2
3			0

D'	1	2	3
1	0	8	8
2		0	3
3			0

$$d'_{12} = 8 - \tfrac{1}{2}(0 + 8 - 8) = 8$$
$$d'_{13} = 8 - \tfrac{1}{2}(0 + 4 - 4) = 8$$
$$d'_{23} = 8 - \tfrac{1}{2}(8 + 4 - 2) = 3$$

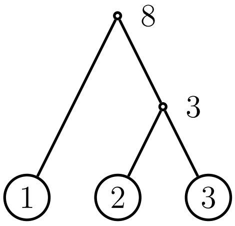


Abbildung 7.22: Gegenbeispiel: D nicht additiv, aber D' ultrametrisch

.

Fall, auch wenn dies in vielen Lehrbüchern und Skripten fälschlicherweise behauptet wird. Wir geben hierfür ein Gegenbeispiel in Abbildung 7.22. Man sieht leicht, dass D' ultrametrisch D ist hingegen nicht additiv, weil $d(1,2) = 8$, aber

$$d(1,3) + d(3,2) = 4 + 2 = 6 < 8$$

gilt. Im Baum muss also der Umweg über 3 größer sein als der direkte Weg, was nicht sein kann (siehe auch Abbildung 7.23), denn im additiven Baum gilt die normale Dreiecksungleichung (siehe Lemma 7.17).

Dennoch können wir mit einer weiteren Zusatzbedingung dafür sorgen, dass das Lemma in der von uns gewünschten Weise retten können.

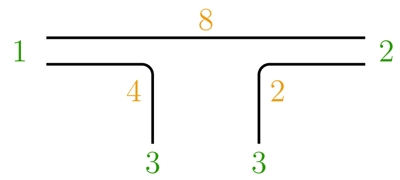


Abbildung 7.23: Gegenbeispiel: „Unmöglicher“ additiver Baum

Lemma 7.20 *Sei D eine Distanzmatrix und sei d_{ij} ein maximaler Eintrag von D. Weiter sei D' durch $d'_{k\ell} = d_{ij} - \frac{1}{2}(d_{ik} + d_{i\ell} - d_{k\ell})$ definiert. Wenn D' eine ultrametrische Matrix ist und wenn für jedes Blatt b im zugehörigen ultrametrischen Baum $T(D')$ für das Gewicht γ der zu b inzidenten Kante gilt: $\gamma \geq (d_{ij} - d_{bi})$, dann ist D additiv.*

Beweis: Sei $T' := T(D')$ ein ultrametrischer Baum für D'. Weiter sei $(v, w) \in E(T')$ und es seien p, q, r, s Blätter von T', so dass $\mathrm{lca}(p, q) = v$ und $\mathrm{lca}(r, s) = w$. Dies ist in Abbildung 7.24 illustriert. Hierbei ist q zweimal angegeben ist, da a priori nicht klar ist, ob q ein Nachfolger von w ist oder nicht. Wir definieren dann das

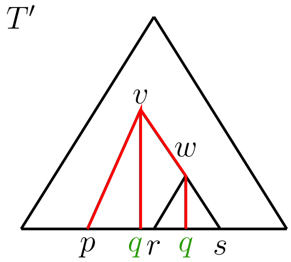


Abbildung 7.24: Skizze: Kante (v, w) in T'

Kantengewicht von (v, w) durch:

$$\gamma(v, w) = d'_{pq} - d'_{rs}.$$

Damit ergibt sich für

$$\begin{aligned} d_{T'}(k, \ell) &= 2 \cdot d'_{k,\ell} \\ &= 2(d_{ij} - \frac{1}{2}(d_{ik} + d_{i\ell} - d_{k\ell})) \\ &= 2d_{ij} - d_{ik} - d_{i\ell} + d_{k\ell} \end{aligned}$$

Wenn wir jetzt für jedes Blatt b das Gewicht der inzidenten Kante um $d_{ij} - d_{bi}$ erniedrigen, erhalten wir einen neuen additiven Baum T. Dann nach Voraussetzung, das Gewicht einer solchen zu b inzidenten Kante größer als $d_{ij} - d_{bi}$ ist, bleiben die Kantengewichte von T positiv. Weiterhin gilt in T:

$$\begin{aligned} d_T(k,\ell) &= d_{T'}(k,\ell) - (d_{ij} - d_{ik}) - (d_{ij} - d_{i\ell}) \\ &= (2d_{ij} - d_{ik} - d_{i\ell} + d_{k\ell}) - d_{ij} + d_{ik} - d_{ij} + d_{i\ell} \\ &= d_{k\ell}. \end{aligned}$$

Somit ist T ein additiver Baum für D und das Lemma ist bewiesen. ∎

Gehen wir noch einmal zurück zu unserem Gegenbeispiel, das besagte, dass die Ultrametrik von D' nicht ausreicht um die Additivität von D zu zeigen. Wir schauen einmal was hier beim Kürzen der Gewichte von zu Blättern inzidenten Kanten passieren kann. Dies ist in Abbildung 7.25 illustriert. Wir sehen hier, dass der Baum zwar die gewünschten Abstände besitzt, jedoch negative Kantengewichte erzeugt, die bei additiven Bäumen nicht erlaubt sind.

D	1	2	3
1	0	8	4
2		0	2
3			0

D'	1	2	3
1	0	8	8
2		0	3
3			0

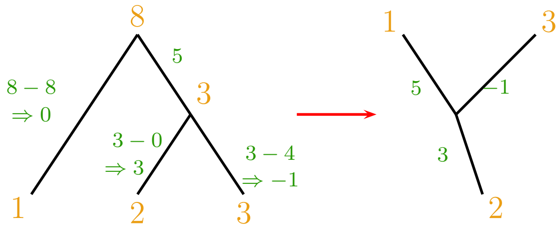


Abbildung 7.25: Gegenbeispiel: D nicht additiv und D' ultrametrisch (Fortsetzung)

7.3.3 Algorithmus zur Erkennung additiver Matrizen

Aus der Charakterisierung der additiven Matrizen mit Hilfe ultrametrischer Matrizen lässt sich der folgende Algorithmus zur Erkennung additiver Matrizen und der Konstruktion zugehöriger additiver Bäume herleiten. Dieser ist in Abbildung 7.26 aufgelistet. Es lässt sich leicht nachrechnen, dass dieser Algorithmus eine Laufzeit von $O(n^2)$ besitzt.

Wir müssen uns nur noch kurz um die Korrektheit kümmern. Nach Lemma 7.19 wissen wir, dass in Schritt 2 die richtige Entscheidung getroffen wird. Nach Lemma 7.20 wird auch im Schritt 4 die richtige Entscheidung getroffen. Also ist der Algorithmus korrekt. Fassen wir das Ergebnis noch zusammen.

Theorem 7.21 *Es kann in Zeit $O(n^2)$ entschieden werden, ob eine gegebene $n \times n$-Distanzmatrix additiv ist oder nicht. Falls die Matrix additiv ist, kann der zugehörige additive Baum in Zeit $O(n^2)$ konstruiert werden.*

1. Konstruiere aus der gegebenen $n \times n$-Matrix D eine neue $n \times n$-Matrix D' mittels $d'_{k\ell} = d_{ij} - \frac{1}{2}(d_{ik} + d_{i\ell} - d_{k\ell})$, wobei d_{ij} ein maximaler Eintrag von D ist.

2. Teste D' auf Ultrametrik. Ist D' nicht ultrametrisch, dann ist D nicht additiv.

3. Konstruiere den zu D' gehörigen ultrametrischen Baum T'.

4. Teste, ob sich die Kantengewichte für jedes Blatt b um $d_{ij} - d_{ib}$ erniedrigen lässt. Falls dies nicht geht, ist D nicht additiv, andernfalls erhalten wir einen additiven Baum T für D.

Abbildung 7.26: Algorithmus: Erkennung additiver Matrizen

7.3.4 4-Punkte-Bedingung

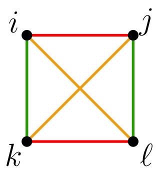
Zum Abschluss wollen wir noch eine andere Charakterisierung von additiven Matrizen angeben, die so genannte *4-Punkte-Bedingung von Buneman*.

Lemma 7.22 (Bunemans 4-Punkte-Bedingung) *Eine $n \times n$-Distanzmatrix D ist genau dann additiv, wenn für je vier Punkte $i, j, k, \ell \in [1 : n]$ mit*

$$d_{i\ell} + d_{jk} = \min\{d_{ij} + d_{k\ell}, d_{ik} + d_{j\ell}, d_{i\ell} + d_{jk}\}$$

gilt, dass $d_{ij} + d_{k\ell} = d_{ik} + d_{j\ell}$.

Diese 4-Punkte-Bedingung ist in Abbildung 7.27 noch einmal illustriert



$$d_{i\ell} + d_{jk} = \min\{d_{ij} + d_{k\ell}, d_{ik} + d_{i\ell}, d_{i\ell} + d_{jk}\}$$
$$\Rightarrow \quad d_{ij} + d_{k\ell} = d_{ik} + d_{j\ell}$$

Abbildung 7.27: Skizze: 4-Punkte-Bedingung

Beweis: **$\Rightarrow$:** Sei T ein additiver Baum für D. Wir unterscheiden jetzt drei Fälle, je nachdem, wie die Pfade in T verlaufen.

Fall 1: Im ersten Fall nehmen wir an, die Pfade von i nach j und k nach ℓ knotendisjunkt sind. Dies ist in Abbildung 7.28 illustriert. Dann ist aber $d_{i\ell} + d_{jk}$ nicht minimal und somit ist nichts zu zeigen.

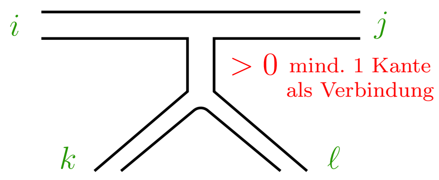


Abbildung 7.28: Skizze: Die Pfade $i \to j$ und $k \to \ell$ sind knotendisjunkt

Fall 2: Im zweiten Fall nehmen wir an, die Pfade von i nach k und j nach ℓ knotendisjunkt sind. Dies ist in Abbildung 7.29 illustriert. Dann ist jedoch ebenfalls $d_{i\ell} + d_{jk}$ nicht minimal und somit ist nichts zu zeigen.

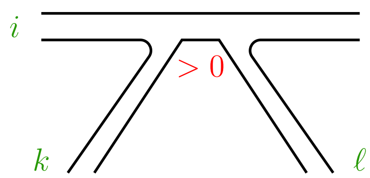


Abbildung 7.29: Skizze: Die Pfade $i \to k$ und $j \to \ell$ sind knotendisjunkt

Fall 3: Im dritten und letzten Fall nehmen wir an, die Pfade von i nach ℓ und j nach k kantendisjunkt sind und sich daher höchstens in einem Knoten schneiden. Dies ist in Abbildung 7.30 illustriert. Nun gilt jedoch, wie man leicht der Abbildung 7.30

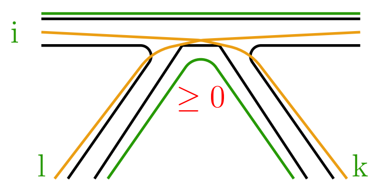


Abbildung 7.30: Skizze: Die Pfade $i \to \ell$ und $j \to k$ sind kantendisjunkt

entnehmen kann:

$$d_{ij} = d_{ik} + d_{j\ell} - d_{k\ell}$$

und somit

$$d_{ij} + d_{k\ell} = d_{ik} + d_{j\ell}.$$

$\Leftarrow$: Betrachte D' mit $d'_{k\ell} := d_{ij} - \frac{1}{2}(d_{ik} + d_{i\ell} - d_{k\ell})$, wobei d_{ij} ein maximaler Eintrag von D ist. Es genügt zu zeigen, dass D' ultrametrisch ist.

Betrachte $p, q, r \in [1:n]$ mit $d'_{pq} \leq d'_{pr} \leq d'_{qr}$. Es ist dann zu zeigen: $d'_{pr} = d'_{qr}$. Aus $d'_{pq} \leq d'_{pr} \leq d'_{qr}$ folgt dann:

$$2d_{ij} - d_{ip} - d_{iq} + d_{pq} \;\leq\; 2d_{ij} - d_{ip} - d_{ir} + d_{pr} \;\leq\; 2d_{ij} - d_{iq} - d_{ir} + d_{qr}.$$

Daraus folgt:

$$d_{pq} - d_{ip} - d_{iq} \;\leq\; d_{pr} - d_{ip} - d_{ir} \;\leq\; d_{qr} - d_{iq} - d_{ir}$$

und weiter

$$d_{pq} + d_{ir} \;\leq\; d_{pr} + d_{iq} \;\leq\; d_{qr} + d_{ip}.$$

Aufgrund der 4-Punkt-Bedingung folgt unmittelbar

$$d_{pr} + d_{iq} = d_{qr} + d_{ip}.$$

Nach Addition von $(2d_{ij} - d_{ir})$ folgt:

$$(2d_{ij} - d_{ir}) + d_{pr} + d_{iq} = (2d_{ij} - d_{ir}) + d_{qr} + d_{ip}.$$

Nach kurzer Rechnung folgt:

$$2d_{ij} - d_{ir} - d_{ip} + d_{pr} = 2d_{ij} - d_{ir} - d_{iq} + d_{qr}$$

und somit gilt

$$2d'_{pr} = 2d'_{qr}.$$

Also ist D' nach Lemma 7.4 ultrametrisch und somit ist nach Lemma7.20 D additiv. ■

Mit Hilfe dieser Charakterisierung werden wir noch zeigen, dass es Matrizen gibt, die eine Metrik induzieren, aber keinen additiven Baum besitzen. Dieses Gegenbeispiel

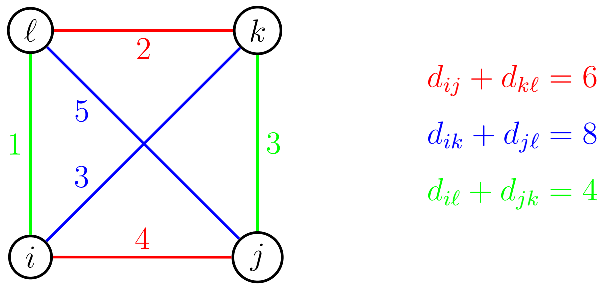


Abbildung 7.31: Gegenbeispiel: Metrische Matrix, die nicht additiv ist

ist in Abbildung 7.31 angegeben. Man sieht leicht, dass die 4-Punkte-Bedingung nicht gilt und somit die Matrix nicht additiv ist. Man überprüft leicht, dass für jedes Dreieck die Dreiecksungleichung gilt, die Abstände also der Definition einer Metrik genügen.

7.3.5 Charakterisierung kompakter additiver Bäume

In diesem Abschnitt wollen wir eine schöne Charakterisierung kompakter additiver Bäume angeben. Zunächst benötige wir noch einige grundlegenden Definitionen aus der Graphentheorie.

Definition 7.23 *Sei $G = (V, E)$ ein ungerichteteter Graph. Ein Teilgraph $G' \subset G$ heißt* aufspannend*, wenn $V(G') = V(G)$ und G' zusammenhängend ist.*

Der aufspannende Teilgraph enthält also alle Knoten des aufgespannten Graphen. Nun können wir den Begriff eines Spannbaumes definieren.

Definition 7.24 *Sei $G = (V, E)$ ein ungerichteteter Graph. Ein Teilgraph $G' \subset G$ heißt* Spannbaum*, wenn G' ein aufspannender Teilgraph von G ist und G' ein Baum ist.*

Für gewichtete Graphen brauchen wir jetzt noch das Konzept eines minimalen Spannbaumes.

Definition 7.25 *Sei $G = (V, E, \gamma)$ ein gewichteter ungerichteteter Graph und T ein Spannbaum von G. Das* Gewicht des Spannbaumes *T von G ist definiert durch*

$$\gamma(T) := \sum_{e \in E(T)} \gamma(e).$$

Ein Spannbaum T für G heißt minimal*, wenn er unter allen möglichen Spannbäumen für G minimales Gewicht besitzt, d.h.*

$$\gamma(T) \leq \min \{\gamma(T') \,:\, T' \textit{ ist ein Spannbaum von } G\}.$$

Im Folgenden werden wir der Kürze wegen einen minimalen Spannbaum oft auch mit MST (engl. *minimum spanning tree*) abkürzen.

Definition 7.26 *Sei D eine $n \times n$-Distanzmatrix. Dann ist $G(D) = (V, E)$ der zu D* gehörige gewichtete Graph, *wobei*

$$\begin{aligned} V &= [1 : n], \\ E &= \binom{V}{2} := \{\{v, w\} : v \neq w \in V\}, \\ \gamma(v, w) &= D(v, w). \end{aligned}$$

Nach diesen Definitionen kommen wir zu dem zentralen Lemma dieses Abschnittes, der kompakte additive Bäume charakterisiert.

Theorem 7.27 *Sei D eine $n \times n$-Distanzmatrix. Besitzt D einen kompakten additiven Baum T, dann ist T der minimale Spannbaum von $G(D)$ und ist eindeutig bestimmt.*

Beweis: Sei T ein kompakter additiver Baum für D (dieser muss natürlich nicht gewurzelt sein). Wir betrachten ein Knotenpaar (x, y), das durch keine Kante in T

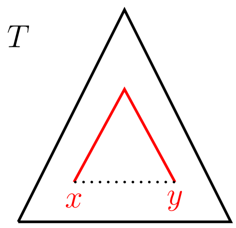


Abbildung 7.32: Skizze: Pfad von x nach y in T

verbunden ist. Sei $p = (v_0, v_1, \ldots, v_k)$ mit $v_0 = x$ und $v_k = y$ sowie $k \geq 2$ der Pfad von x nach y in T. $\gamma(p)$ entspricht $D(x, y)$, weil T ein additiver Baum für D ist. Da nach Definition einer Distanzmatrix alle Kantengewichte positiv sind und der Pfad aus mindestens zwei Kanten besteht, ist $\gamma(v_{i-1}, v_i) < D(x, y) = \gamma(x, y)$ für alle Kanten (v_{i-1}, v_i) des Pfades p.

Sei jetzt T' ein minimaler Spannbaum von $G(D)$. Wir nehmen jetzt an, dass die Kante (x, y) eine Kante des minimalen Spannbaumes ist, d.h. $(x, y) \in E(T')$. Somit verbindet die Kante (x, y) zwei Teilbäume von T'. Dies ist in Abbildung 7.32 illustriert.

Da (x, y) keine Kante im Pfad von x nach y im kompakten additiven Baum von T ist, verbindet der Pfad p die beiden durch Entfernen der Kante (x, y) separierten

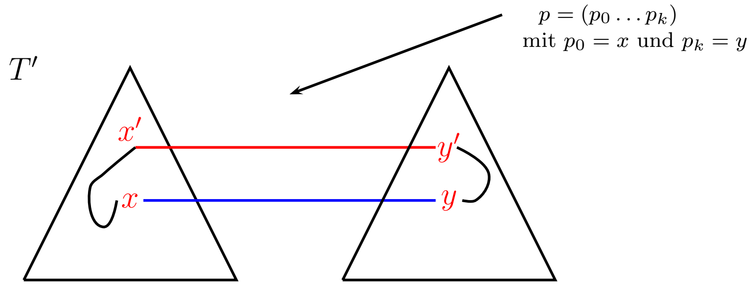


Abbildung 7.33: Skizze: Spannbaum T' von $D(G)$

Teilbäume des minimalen Spannbaumes T'. Sei (x', y') die erste Kante auf dem Pfad p von x nach y, die nicht innerhalb einer der beiden separierten Spannbäume verläuft. Wie wir oben gesehen haben, gilt $\gamma(x', y') < \gamma(x, y)$.

Wir konstruieren jetzt einen neuen Spannbaum T'' für $G(D)$ wie folgt:

$$\begin{aligned} V(T'') &= V(T') = V \\ E(T'') &= (E(T') \setminus \{\{x, y\}\}) \cup \{\{x', y\}\} \end{aligned}$$

Für das Gewicht des Spannbaumes T'' gilt dann:

$$\begin{aligned} \gamma(T'') &= \sum_{e \in E(T'')} \gamma(e) \\ &= \sum_{e \in E(T')} \gamma(e) - \gamma(x, y) + \gamma(x', y') \\ &= \sum_{e \in E(T')} \gamma(e) + \underbrace{(\gamma(x', y') - \gamma(x, y))}_{<0} \\ &< \gamma(T'). \end{aligned}$$

Somit ist $\gamma(T'') < \gamma(T)$ und dies liefert den gewünschten Widerspruch zu der Annahme, dass T' ein minimaler Spannbaum von $G(D)$ ist.

Somit gilt für jedes Knotenpaar (x, y) mit $(x, y) \notin E(T)$, dass dann die Kante (x, y) nicht im minimalen Spannbaum von $G(D)$ sein kann, d.h. $(x, y) \notin T'$. Also ist eine Kante, die sich nicht im kompakten additiven Baum befindet, auch keine Kante des Spannbaums.

Dies gilt sogar für zwei verschiedene minimale Spannbäume für $G(D)$. Somit kann es nur einen minimalen Spannbaum geben. ■

7.3.6 Konstruktion kompakter additiver Bäume

Die Information aus dem vorherigen Lemma kann man dazu ausnutzen, um einen effizienten Algorithmus zur Erkennung kompakter additiver Matrizen zu entwickeln. Sei D die Matrix, für den der kompakte additive Baum konstruiert werden soll, und somit $G(D) = (V, E, \gamma)$ der gewichtete Graph, für den der minimale Spannbaum berechnet werden soll.

Wir beginnen mit der Knotenmenge $V' = \{v\}$ für eine beliebiges $v \in V$ und der leeren Kantenmenge $E' = \emptyset$. Der Graph (V', E') soll letztendlich der gesuchte minimale Spannbaum werden. Wir verwenden hier Prims Algorithmus zur Konstruktion eines minimalen Spannbaumes, der wieder ein Greedy-Algorithmus sein wird. Wir versuchen die Menge V' in jedem Schritt um einen Knoten aus $V \setminus V'$ zu erweitern. Von allen solchen Kandidaten wählen wir eine Kante minimalen Gewichtes. Dies ist schematisch in Abbildung 7.34 dargestellt.

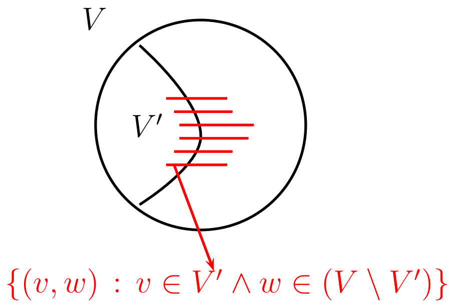


Abbildung 7.34: Skizze: Erweiterung von V'

Den Beweis, dass wir hiermit einen minimalen Spannbaum konstruieren, wollen wir an dieser Stelle nicht führen. Wir verweisen hierzu auf die einschlägige Literatur. Den formalen Algorithmus haben wir in Abbildung 7.35 angegeben. Bis auf die blauen Zeilen ist dies genau der Pseudo-Code für Prims Algorithmus zur Konstruktion eines minimalen Spannbaums.

Die blaue for-Schleife ist dazu da, um zu testen, ob der konstruierte minimale Spannbaum wirklich ein kompakter additiver Baum für D ist. Zum einen setzen wir den konstruierten Abstand d_T für den konstruierten minimalen Spannbaum. Der nachfolgende Test überprüft, ob dieser Abstand d_T mit der vorgegebenen Distanzmatrix γ übereinstimmt.

Wir müssen uns jetzt nur noch die Laufzeit überlegen. Die while-Schleife wird genau $n = |V|$ Mal durchlaufen. Danach ist $V' = V$. Die Minimumsbestimmung lässt sich sicherlich in Zeit $O(n^2)$ erledigen, da maximal n^2 Kanten zu durchsuchen sind. Daraus folgt eine Laufzeit von $O(n^3)$.

```
Modified_Prim (V, E, γ)
{
    E' := ∅;
    V' := {v} für ein beliebiges v ∈ V;
    /* (V', E') wird am Ende der minimale Spannbaum sein */
    while (V' ≠ V)
    {
        Sei e = (x, y), so dass γ(x, y) = min {γ(v, w) : v ∈ V' ∧ w ∈ V \ V'};
        /* oBdA sei y ∈ V' */
        E' := E' ∪ {e};
        V' := V' ∪ {y};
        for all (v ∈ V');
        {
            d_T(v, y) := d_T(v, x) + γ(x, y);
            if (d_T(v, y) ≠ γ(v, y)) reject;
        }
    }
    return (V', E');
}
```

Abbildung 7.35: Algorithmus: Konstruktion eines kompakten additiven Baumes

Implementiert man Prims-Algorithmus ein wenig geschickter, z.B. mit Hilfe von Priority-Queues, so lässt sich die Laufzeit auf $O(n^2)$ senken. Für die Details verweisen wir auch hier auf die einschlägige Literatur. Auch die blaue Schleife kann insgesamt in Zeit $O(n^2)$ implementiert werden, da jede for-Schleife maximal n-mal durchlaufen wird und die for-Schleife selbst maximal n-mal ausgeführt wird

Theorem 7.28 *Sei D eine $n \times n$-Distanzmatrix. Dann lässt sich in Zeit $O(n^2)$ entscheiden, ob ein kompakter additiver Baum für D existiert. Falls dieser existiert, kann dieser ebenfalls in Zeit $O(n^2)$ konstruiert werden.*

7.4 Perfekte binäre Phylogenie

In diesem Abschnitt wollen wir uns jetzt um charakterbasierte Methoden kümmern. Wir beschränken uns hier auf den Spezialfall, dass die Charaktere binär sind, d.h. nur zwei verschiedene Werte annehmen können.

7.4.1 Charakterisierung perfekter Phylogenie

Zunächst benötigen wir noch ein paar grundlegende Definitionen, um das Problem der Konstruktion phylogenetischer Bäume formulieren zu können.

Definition 7.29 *Eine* binäre Charaktermatrix M *ist eine binäre* $n \times m$*-Matrix. Ein* phylogenetischer Baum T *für eine binäre* $n \times m$*-Charaktermatrix ist ein ungeordneter gewurzelter Baum mit genau* n *Blättern, so dass:*

1. *Jedes der* n *Objekte aus* $[1 : n]$ *markiert genau eines der Blätter;*
2. *Jeder der* m *Charaktere aus* $[1 : m]$ *markiert genau eine Kante;*
3. *Für jedes Objekt* $p \in [1 : n]$ *gilt für die Menge* C_T *der Kantenmarkierungen auf dem Pfad von der Wurzel von* T *zu dem Blatt mit Markierung* p*, dass* $C_T = \{c : M_{c,p} = 1\}$.

In Abbildung 7.36 ist eine binäre Charaktermatrix samt seines zugehörigen phylogenetischen Baumes angegeben.

M	1	2	3	4	5	6	7
1	1	0	1	0	0	1	0
2	1	0	1	0	1	0	1
3	1	0	1	0	1	0	0
4	1	0	0	0	0	0	0
5	0	1	1	0	0	0	0
6	0	1	0	0	0	0	0

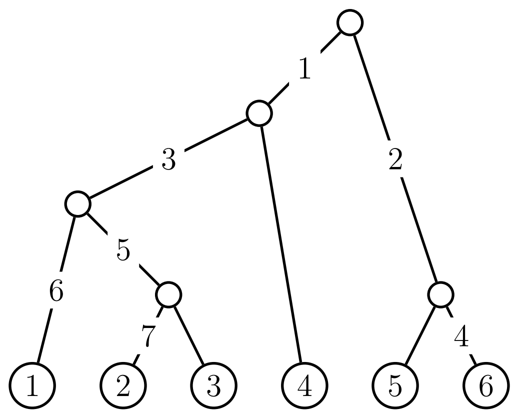


Abbildung 7.36: Beispiel: binäre Charaktermatrix und zugehöriger phylogenetischer Baum

Somit können wir das Problem der perfekten Phylogenie formulieren.

PERFEKTE PHYLOGENIE

Eingabe: Eine binäre $n \times m$-Charaktermatrix M.
Ausgabe: Ein phylogenetischer Baum T für M.

Zunächst benötigen wir noch eine weitere Notation bevor wir uns der Lösung des Problems zuwenden können.

Notation 7.30 *Sei M eine binäre $n \times m$-Charaktermatrix, dann umfasst die Menge $O_j = \{i \in [1:n] : M_{i,j} = 1\}$ die Objekte, die den Charakter j besitzen.*

Wir geben zunächst wieder eine Charakterisierung an, wann eine binäre Charaktermatrix überhaupt einen phylogenetischen Baum besitzt.

Theorem 7.31 *Eine binäre $n \times m$-Charaktermatrix M besitzt genau dann einen phylogenetischen Baum, wenn für jedes Paar $i, j \in [1:n]$ entweder $O_i \cap O_j = \emptyset$ oder $O_i \subset O_j$ oder $O_i \supset O_j$ gilt.*

Beweis: **$\Rightarrow$:** Sei T ein phylogenetischer Baum für M. Sei e_i bzw. e_j die Kante in T mit der Markierung i bzw. j. Wir unterscheiden jetzt vier Fälle, je nachdem, wie sich die Kanten e_i und e_j zueinander im Baum T verhalten.

Fall 1: Wir nehmen zuerst an, dass $e_i = e_j$ ist (siehe auch Abbildung 7.37). In diesem Fall gilt offensichtlich $O_i = O_j$.

Fall 2: Nun nehmen wir an, dass sich im Teilbaum vom unteren Knoten vom e_i die Kante e_j befindet (siehe auch Abbildung 7.37). Also sind alle Nachfahren des unteren Knotens von e_j auch Nachfahren des unteren Knoten von e_i und somit gilt $O_j \subset O_i$.

Fall 3: Nun nehmen wir an, dass sich im Teilbaum vom unteren Knoten vom e_j die Kante e_i befindet (siehe auch Abbildung 7.37). Also sind alle Nachfahren des unteren Knotens von e_i auch Nachfahren des unteren Knoten von e_j und somit gilt $O_i \subset O_j$.

Fall 4: Der letzte verbleibende Fall ist, dass keine Kante Nachfahre eine anderen Kante ist (siehe auch Abbildung 7.37). In diesem Fall gilt offensichtlich $O_i \cap O_j = \emptyset$.

$\Leftarrow$: Wir müssen jetzt aus der Matrix M einen phylogenetischen Baum konstruieren. Zuerst nehmen wir ohne Beschränkung der Allgemeinheit an, dass die Spalten der Matrix M interpretiert als Binärzahlen absteigend sortiert sind. Dabei nehmen wir an, dass jeweils in der ersten Zeile das höchstwertigste Bit und in der letzten Zeile das niederwertigste Bit steht. Wir machen uns dies noch einmal anhand unserer Beispiel aus der Einleitung klar und betrachten dazu Abbildung 7.38. Der besseren Übersichtlichkeit wegen, bezeichnen wir die Spalten jetzt mit a–g anstatt mit 1–7.

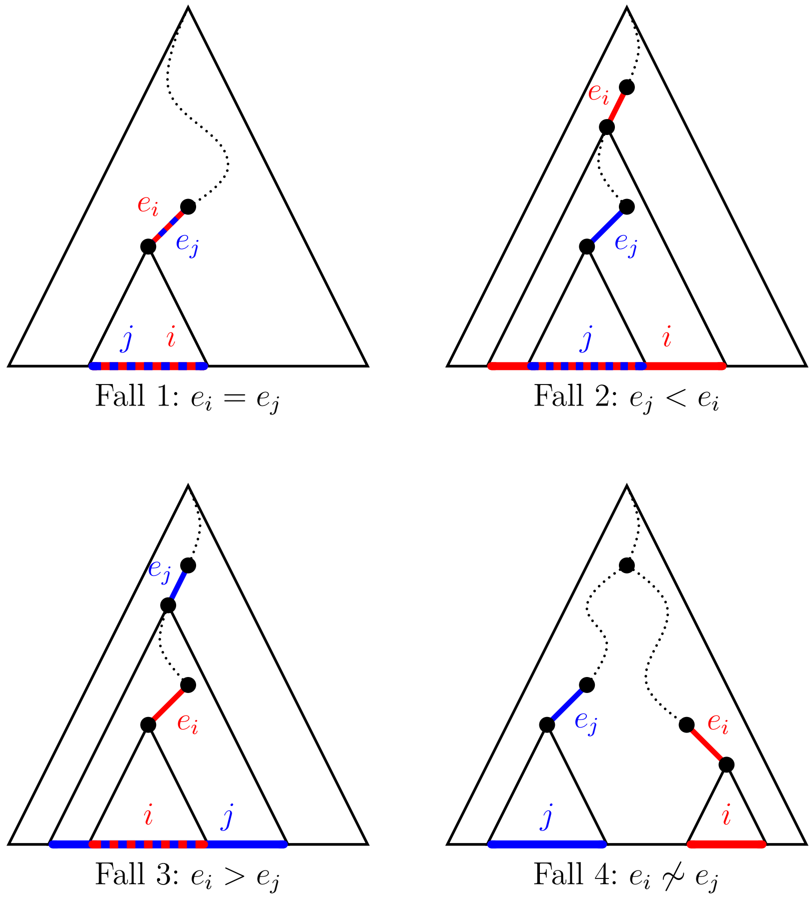


Abbildung 7.37: Skizze: Phylogenetischer Baum für M

M	a	b	c	d	e	f	g
1	1	0	1	0	0	1	0
2	1	0	1	0	1	0	1
3	1	0	1	0	1	0	0
4	1	0	0	0	0	0	0
5	0	1	1	0	0	0	0
6	0	1	0	0	0	0	0

M'	a	c	f	e	g	b	d
1	1	1	1	0	0	0	0
2	1	1	0	1	1	0	0
3	1	1	0	1	0	0	0
4	1	0	0	0	0	0	0
5	0	0	0	0	0	1	1
6	0	0	0	0	0	1	0

Abbildung 7.38: Beispiel: Sortierte Charaktermatrix

Wir betrachten jetzt zwei beliebige Objekte p und q. Sei k der größte gemeinsame Charakter, den sowohl p als auch q besitzt, d.h.

$$k = \max \{j \,:\, M(p, j) = M(q, j) = 1\}.$$

Wir betrachten jetzt einen Charakter $i < k$ von p, d.h. $M(p, i) = 1$. Somit gilt $p \in O_i \cap O_k$ und nach Voraussetzung gilt entweder $O_i \subset O_k$ oder $O_k \subset O_i$. Da die Spalten absteigend sortiert sind, muss die größere Zahl (also die zur Spalte i korrespondierende) mehr 1-Bits besitzen und es gilt somit $O_k \subset O_i$. Da $q \in O_k$ gilt, ist nun auch $q \in O_i$. Analoges gilt für ein $i < k$ mit $M(q,i) = 1$. Damit gilt für alle Charaktere $i \leq k$ (nach der Spaltensortierung), dass entweder beide den Charakter i besitzen oder keiner. Da k der größte Index ist, so dass beide einen Charakter gemeinsam besitzen, gilt für $i > k$, dass aus $M(p,i) = M(q,i)$ folgt, dass beide den Charakter i nicht besitzen, d.h. $M(p,i) = M(q,i) = 0$.

Betrachten wir also zwei Objekte (also zwei Zeilen in der spaltensortierten Charaktermatrix), dann besitzen zu Beginn entweder beide einen Charakter gemeinsam oder keinen. Sobald wir einen Charakter finden, der beide Objekte unterscheidet, so finden wir später keine gemeinsamen Charaktere mehr.

Mit Hilfe dieser Eigenschaft können wir jetzt einen phylogenetischen Baum konstruieren. Dazu betrachten wir die Abbildung $p \mapsto s_{p,1} \cdots s_{p,\ell}\$$, wobei wir jedem Objekt eine Zeichenkette über $[1:m]$ zuordnen, so dass jedes Symbol aus $[1:m]$ maximal einmal auftaucht und ein $i \in [1:m]$ nur dann auftaucht, wenn p den entsprechenden Charakter besitzt, also wenn $M(p,i) = 1$. Die Reihenfolge ergibt sich dabei aus der Spaltensortierung der Charaktermatrix.

Für unser Beispiel ist diese Zuordnung in der Abbildung 7.39 angegeben.

$$\begin{aligned} s_1 &= acf\$ \\ s_2 &= aceg\$ \\ s_3 &= ace\$ \\ s_4 &= a\$ \\ s_5 &= bd\$ \\ s_6 &= b\$ \end{aligned}$$

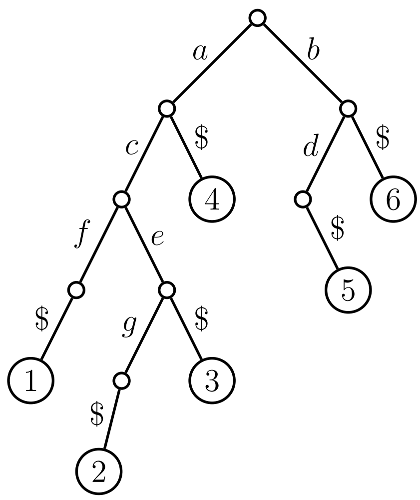


Abbildung 7.39: Skizze: Trie für die Zeichenreihen der Charaktermatrix

Wenn wir jetzt für diese Zeichenreihen einen Trie konstruieren (dies entspricht dem Suchmuster-Baum aus dem Aho-Corasick-Algorithmus), so erhalten wir den phylogenetischen Baum für unsere Charaktermatrix M. Wir müssen nur noch Knoten mit nur einem Kind kontrahieren und die Kantenmarkierungen mit dem \$ entfernen.

Aus der Konstruktion folgt, dass alle Blätter die benötigten Kantenmarkierungen auf dem Pfad von der Wurzel zum Blatt besitzen. Nach der Spaltensortierung und der daran anschließenden Diskussion kommt auch jeder Charakter nur einmal als Kantenmarkierung vor. Somit haben wir nachgewiesen, dass der konstruierte Baum ein phylogenetischer Baum ist und der Beweis ist beendet. ■

Aus dem vorherigen Beweis erhalten wir unmittelbar einen Algorithmus zur Berechnung einer perfekten binären Phylogenie, der in Abbildung 7.40 aufgelistet ist.

1. Sortieren der Spalten der binären Charaktermatrix. $O(nm)$
2. Konstruktion der Zeichenreihen $s_1, \ldots, s_n$. $O(nm)$
3. Konstruktion des Tries T für $s_1, \ldots, s_n$. $O(nm)$
4. Kompaktifiziere den Trie T und teste, ob der kompaktifizierte Trie ein phylogenetischer Baum ist. $O(nm)$

Abbildung 7.40: Algorithmus: Konstruktion einer perfekten binäre Phylogenie

Anstatt die Bedingung aus dem Lemma zu überprüfen, konstruieren wir einfach einen kompaktifizierten Trie. Ist dieser ein phylogenetischen Baum, so muss dieser der gesuchte sein. Andernfalls gibt es keinen phylogenetischen Baum für die gegebene binäre Charaktermatrix.

Theorem 7.32 *Für eine binäre $n \times m$-Charaktermatrix M lässt sich in Zeit $O(nm)$ entscheiden, ob sie eine perfekte Phylogenie besitzt. Falls eine perfekte Phylogenie existiert, lässt sich der zugehörige phylogenetische Baum in Zeit $O(nm)$ konstruieren.*

7.4.2 Binäre Phylogenien und Ultrametriken

Nun wollen wir noch zeigen, dass sich das Problem der perfekten Phylogenie auch auf das Problem der Bestimmung eines ultrametrischen Baumes zurückführen lässt. Zunächst definieren wir den Begriff einer phylogenetischen Distanzmatrix

Definition 7.33 *Sei M eine binäre $n \times m$-Charaktermatrix, dann heißt die $n \times n$-Matrix eine* phylogenetische Distanzmatrix, *wenn*

$$D_M(i,j) = m - |\{c : M(i,c) = M(j,c) = 1\}|.$$

In Abbildung 7.41 ist ein Beispiel für eine binäre Charaktermatrix und der zugehörigen phylogenetischen Distanzmatrix angegeben.

M	1	2	3	4	5	6	7
1	1	0	1	0	0	1	0
2	1	0	1	0	1	0	1
3	1	0	1	0	1	0	0
4	1	0	0	0	0	0	0
5	0	1	1	0	0	0	0
6	0	1	0	0	0	0	0

D_M	1	2	3	4	5	6
1	0	5	5	6	7	7
2		0	4	6	7	7
3			0	6	7	7
4				0	7	7
5					0	6
6						0

Abbildung 7.41: Beispiel: phylogenetische Distanzmatrix einer binären Charaktermatrix

Nun kommen wir zu der gesuchten Charakterisierung.

Lemma 7.34 *Besitzt eine binäre Charaktermatrix eine perfekte Phylogenie, dann ist die zugehörige phylogenetische Distanzmatrix ultrametrisch.*

Beweis: Sei T ein phylogenetischer Baum für eine binäre Charaktermatrix M. Wir konstruieren jetzt daraus einen ultrametrischen Baum T', indem wir zu T Knotenmarkierungen hinzufügen. Markiere die Wurzel mit n. Betrachte nun die Kinder rekursiv. Ist dabei die Kante zu einem Kind mit $k \geq 0$ Charakteren markiert, so besitzen diese alle gemeinsam diesen Charakter im betrachteten Teilbaum und der Maximal-Abstand kann nur noch $m-k$ sein. Mit Hilfe einer Tiefensuche können wir so den Baum durchlaufen und die Knotenmarkierungen vergeben. ■

Aus diesem Beweis ergibt sich der folgende, in Abbildung 7.42 angegebene Algorithmus. Wir schauen uns jetzt nur noch die Details von Schritt 3 genauer an. Alle vier Schritte können jeweils in $O(nm)$ ausgeführt werden. Es ergibt sich somit eine Gesamtlaufzeit von $O(mn^2)$.

Falls M eine perfekte Phylogenie besitzt, so kann diese mit den angegebenen Schritten konstruiert werden. Durch die angegebene Methode zur Konstruktion von D_M liegen alle Werte d_{ij} im Bereich $[0 : m]$. Es wird genau ein Blatt pro Spezies angelegt und im Schritt 3c) wird jeder Charakter eines Blattes zu einer Kante von der Wurzel zum Blatt zugewiesen (ohne Beschränkung der Allgemeinheit sei der Fall ausgeschlossen, dass alle Spezies eine bestimmte Eigenschaft haben). In Schritt 3d) wird dann, dass jeder Charakter nur einmal auftritt. Falls dies gelingt, so ist der entstandene Baum eine perfekte Phylogenie für M.

1. Konstruiere D_M aus M. $O(nm)$
2. Konstruiere den ultrametrischen Baum U für D_M. Lässt sich U nicht konstruieren, dann besitzt M keine perfekte Phylogenie. $O(n^2)$
3. Versuche die Kanten des ultrametrischen Baumes U mit binären Charaktere zu markieren. Lässt sich dies nicht bewerkstelligen, dann besitzt M keine perfekte Phylogenie. $O(n^2m)$
   a) Jedem Blatt (entspricht einer Spezies) wird sein Zeilenvektor aus der gegebenen Matrix M zugewiesen.
   b) Jedem inneren Knoten wird ein $\{0, 1\}$-Vektor der Länge m zugewiesen, der aus der komponentenweisen Multiplikation der Vektoren seiner Kinder entsteht.
   c) Jeder Kante werden die Charaktere zugewiesen, die im Kind auftauchen, aber im Elter nicht.
   d) Wir jeder Buchstabe nur einmal verwendet, entsteht so eine perfekte Phylogenie.

Abbildung 7.42: Algorithmus: Konstruktion eine perfekten binären Phylogenie

Falls M eine perfekte Phylogenie hat, bleibt zu zeigen, dass jeder Charakter nur genau einmal vorkommt (er muss mindestens einmal vorkommen). Die technischen Details überlassen wir dem Leser als Übungsaufgabe und fassen das Ergebnis zusammen.

Theorem 7.35 *Für eine binäre $n \times m$-Charaktermatrix M lässt sich in Zeit $O(mn^2)$ entscheiden, ob sie eine perfekte Phylogenie besitzt. Falls eine perfekte Phylogenie existiert, lässt sich der zugehörige phylogenetische Baum in Zeit $O(mn^2)$ konstruieren.*

7.5 Sandwich Probleme

Hauptproblem bei den bisher vorgestellten Verfahren war, dass wir dazu die Distanzen genau wissen mussten, da beispielsweise eine leicht modifizierte ultrametrische Matrix in der Regel nicht mehr ultrametrisch ist. Aus diesem Grund werden wir in diesem Abschnitt einige Problemestellungen vorstellen und lösen, die Fehler in den Eingabedaten modellieren.

7.5.1 Fehlertolerante Modellierungen

Bevor wir zur Problemformulierung kommen, benötigen wir noch einige Notationen, wie wir Matrizen zwischen zwei anderen Matrizen einschachteln können.

Notation 7.36 *Seien M und M' zwei $\times n$-Matrizen, dann gilt $M \leq M'$, wenn $M_{i,j} \leq M'_{i,j}$ für alle $i, j \in [1:n]$ gilt. Für drei Matrizen M, M' und M'' gilt $M \in [M', M'']$, wenn $M' \leq M \leq M''$ gilt.*

Nun können wir die zu untersuchenden Sandwich-Probleme angeben.

ADDITIVES SANDWICH PROBLEM

Eingabe: Zwei $n \times n$-Distanzmatrizen D_ℓ und D_h.
Ausgabe: Eine additive Distanzmatrix $D \in [D_\ell, D_h]$, sofern eine existiert.

ULTRAMETRISCHES SANDWICH PROBLEM

Eingabe: Zwei $n \times n$-Distanzmatrizen D_ℓ und D_h.
Ausgabe: Eine ultrametrische Distanzmatrix $D \in [D_\ell, D_h]$, sofern eine existiert.

Im Folgenden bezeichnet $\|M\|$ eine Norm einer Matrix. Hierbei wird diese Norm jedoch nicht eine Abbildungsnorm der durch M induzierten Abbildung sein, sondern wir werden Normen verwenden, die eine $n \times n$-Matrix als einen Vektor mit n^2 Einträgen interpretiert. Beispielsweise können wir die so genannten p-Normen verwenden:

$$\begin{aligned} \|D\|_p &= \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |M_{i,j}|^p \right)^{1/p}, \\ \|D\|_\infty &= \max \left\{ |M_{i,j}| : i, j \in [1:n] \right\}. \end{aligned}$$

ADDITIVES APPROXIMATIONSPROBLEM

Eingabe: Eine $n \times n$-Distanzmatrix D.
Ausgabe: Eine additive Distanzmatrix D', die $\|D - D'\|$ minimiert.

Ultrametrisches Approximationsproblem

Eingabe: Eine $n \times n$-Distanzmatrix D.
Ausgabe: Eine ultrametrische Distanzmatrix D', die $\|D - D'\|$ minimiert.

Für die weiteren Untersuchungen benötigen wir noch einige Notationen.

Notation 7.37 *Sei M eine $n \times n$-Matrix, dann ist* $\mathrm{MAX}(M)$ *der maximale Eintrag von M, d.h.* $\mathrm{MAX}(M) = \max\{M_{i,j} \,:\, i, j \in [1:n]\}$.

Sei T ein additiver Baum mit n markierten Knoten (mit Markierungen aus $[1:n]$) und $d_T(i,j)$ der durch den additiven Baum induzierte Abstand zwischen den Knoten mit Markierung i und j, dann ist $\mathrm{MAX}(T) = \max\{d_T(i,j) \,:\, i, j \in [1:n]\}$.

Sei M eine $n \times n$-Matrix und T ein additiver Baum mit n markierten Knoten, dann schreiben wir $T \leq M$ bzw. $T \geq M$, wenn $d_T(i,j) \leq M_{i,j}$ bzw. $d_T(i,j) \geq M_{i,j}$ für alle $i, j \in [1:n]$ gilt.

Für zwei Matrizen M und M' sowie einen additiven Baum T gilt $T \in [M, M']$, wenn $M \leq T \leq M'$ gilt.

Wir wollen noch einmal kurz daran erinnern, wie man aus einem ultrametrischen Baum $T = (V, E, \mu)$ mit der Knotenmarkierung $\mu : V \to \mathbb{R}_+$ einen additiven Baum $T = (V, E, \gamma)$ mit der Kantengewichtsfunktion $\gamma : E \to \mathbb{R}_+$ erhält, indem man γ wie folgt definiert:

$$\forall (v, w) \in E : \gamma(v, w) := \frac{\mu(v) - \mu(w)}{2} > 0.$$

Somit können wir ultrametrische Bäume immer auch als additive Bäume auffassen.

7.5.2 Eine einfache Lösung

In diesem Abschnitt wollen wir zuerst eine einfache Lösung angeben, um die Ideen hinter der Lösung zu erkennen. Im nächsten Abschnitt werden wir dann eine effizientere Lösung angeben, die von der Idee her im Wesentlichen gleich sein wird, aber dort nicht so leicht bzw. kaum zu erkennen sein wird.

Zuerst zeigen wir, dass wenn es einen ultrametrischen Baum gibt, der der Sandwich-Bedingung genügt, es auch einen ultrametrischen Baum gibt, dessen Wurzelmarkierung gleich dem maximalem Eintrag der Matrix D_ℓ ist. Dies liefert einen ersten Ansatzpunkt für einen rekursiven Algorithmus.

Lemma 7.38 *Seien D_ℓ und D_h zwei $n \times n$-Distanzmatrizen. Wenn ein ultrametrischer Baum T mit $T \in [D_\ell, D_h]$ existiert, dann gibt es einen ultrametrischen Baum T' mit $T' \in [D_\ell, D_h]$ und* $\mathrm{MAX}(T') = \mathrm{MAX}(D_\ell)$.

Beweis: Wir beweisen die Behauptung durch Widerspruch. Wir nehmen also an, dass es keinen ultrametrischen Baum $T' \in [D_\ell, D_h]$ mit $\mathrm{MAX}(T') = \mathrm{MAX}(D_\ell)$ gibt.

Sei $T' \in [D_\ell, D_h]$ ein ultrametrischer Baum, der $s := \mathrm{MAX}(T') - \mathrm{MAX}(D_\ell)$ minimiert. Nach Voraussetzung gibt es mindestens einen solchen Baum und nach unserer Annahme für den Widerspruchsbeweis ist $s > 0$.

Wir konstruieren jetzt aus T' einen neuen Baum T'', indem wir nur die Kantengewichte der Kanten in T'' ändern, die zur Wurzel von T' bzw. T'' inzident sind. Zuerst definieren wir $\alpha := \frac{1}{2}\min\{\gamma(e), s\} > 0$. Wir bemerken, dass dann $\alpha \leq \frac{\gamma(e)}{2}$ und $\alpha \leq \frac{s}{2}$ gilt.

Sei also e ein Kante, die zur Wurzel von T'' inzident ist. Dann setzen wir

$$\gamma''(e) = \gamma'(e) - \alpha.$$

Hierbei bezeichnet γ' bzw. γ'' die Kantengewichtsfunktion von T' bzw. T''. Zuerst halten wir fest, dass die Kantengewichte der Kanten, die zur Wurzel inzident sind, weiterhin positiv sind, da

$$\gamma(e) - \alpha \;\geq\; \gamma(e) - \frac{\gamma(e)}{2} \;=\; \frac{\gamma(e)}{2} \;>\; 0.$$

Da wir Kantengewichte nur reduzieren, gilt offensichtlich weiterhin $T'' \leq D_h$. Wir müssen also nur noch zeigen, dass auch $D_\ell \leq T''$ weiterhin gilt. Betrachten wir hierzu zwei Blätter v und w in T'' und die Wurzel $r(T'')$ von T''. Wir unterscheiden jetzt zwei Fälle, je nachdem, ob der kürzeste Weg von v nach w über die Wurzel führt oder nicht.

Fall 1 ($\mathbf{lca}(v, w) \neq r(T'')$): Dann wird der Abstand von $d_{T''}$ gegenüber $d_{T''}$ für diese Blätter nicht verändert und es gilt:

$$d_{T''}(v, w) = d_{T'}(v, w) \geq D_\ell(v, w).$$

Fall 2 ($\mathbf{lca}(v, w) = r(T'')$): Dann werden die beiden Kantengewichte der Kanten auf dem Weg von v zu w die inzident zur Wurzel sind um jeweils α erniedrigt und wir erhalten:

$$\begin{aligned}
d_{T''}(v,w) &= d_{T'}(v,w) - 2\alpha \\
&\quad \text{da } d_{T'}(v,w) = s + \mathrm{MAX}(D_\ell) \\
&= s + \mathrm{MAX}(D_\ell) - 2\alpha \\
&\quad \text{da } 2\alpha \leq s \\
&\geq s + \mathrm{MAX}(D_\ell) - s \\
&= \mathrm{MAX}(D_\ell) \\
&\geq D_\ell(v,w).
\end{aligned}$$

Also ist $D_\ell \leq T''$. Somit haben wir einen ultrametrischen Baum $T'' \in [D_\ell, D_h]$ konstruiert, für den

$$\begin{aligned}
\mathrm{MAX}(T'') - \mathrm{MAX}(D_\ell) &\leq \mathrm{MAX}(T') - 2\alpha - \mathrm{MAX}(D_\ell) \\
&\quad \text{da } \alpha > 0 \\
&< \mathrm{MAX}(T') - \mathrm{MAX}(D_\ell) \\
&= s.
\end{aligned}$$

gilt. Dies ist offensichtlich ein Widerspruch zur Wahl von T' und das Lemma ist bewiesen. ■

Das vorherige Lemma legt die Definition niedriger ultrametrische Bäume nahe.

Definition 7.39 *Ein ultrametrischer Baum $T \in [D_\ell, D_h]$ heißt* niedrig, *wenn* $\mathrm{MAX}(T) = \mathrm{MAX}(D_\ell)$.

Um uns im weiteren etwas leichter zu tun, benötigen wir einige weitere Notationen.

Notation 7.40 *Sei $T = (V, E)$ ein gewurzelter Baum. Dann bezeichnet $\mathcal{L}(T)$ die Menge der Blätter von T. Für $v \in \mathcal{L}(T)$ bezeichnet*

$$\mathcal{L}(T, v) := \{w \in \mathcal{L}(T) \,:\, \mathrm{lca}(v,w) \neq r(T)\}$$

die Menge aller Blätter die sich im selben Teilbaum der Wurzel von T befinden wie v selbst.

Aus dem vorherigen Lemma und den Notationen folgt jetzt unmittelbar das folgende Korollar.

Korollar 7.41 *Für jeden niedrigen ultrametrischen Baum $T \in [D_\ell, D_h]$ gilt:*

$$\forall x, y \in \mathcal{L}(T) : (D_\ell(x, y) = \mathrm{MAX}(D_\ell)) \Rightarrow (d_T(x, y) = \mathrm{MAX}(D_\ell)).$$

Bevor wir zum zentralen Lemma kommen, müssen wir noch eine weitere grundlegende Definition festlegen.

Definition 7.42 *Seien $D_\ell \leq D_h$ zwei $n \times n$-Matrizen und seien $k, \ell \in [1 : n]$, dann ist der Graph $G_{k,\ell} = (V, E_{k,\ell})$ wie folgt definiert:*

$$\begin{aligned} V &:= [1 : n], \\ E_{k,l} &:= \{(i, j) \,:\, D_h(i, j) < D_\ell(k, \ell)\}. \end{aligned}$$

Mit $C(G_{k,\ell}, v)$ bezeichnen wir die Zusammenhangskomponente von $G_{k,\ell}$, die den Knoten v enthält.

Nun kommen wir zu dem zentralen Lemma für unseren einfachen Algorithmus, das uns beschreibt, wie wir mit Kenntnis des Graphen $G_{k\ell}$ (oder besser dessen Zusammenhangskomponenten) den gewünschte ultrametrischen Baum konstruieren können.

Lemma 7.43 *Seien $D_\ell \leq D_h$ zwei $n \times n$-Matrizen und seien $k, \ell \in [1 : n]$ so gewählt, dass $D_\ell(k, \ell) = \mathrm{MAX}(D_\ell)$. Wenn ein niedriger ultrametrischer Baum $T \in [D_\ell, D_h]$ existiert, dann gibt es auch einen niedrigen ultrametrischen Baum $T' \in [D_\ell, D_h]$ mit*

$$\mathcal{L}(T', v) = V(C(G_{k,\ell}, v))$$

für alle $v \in \mathcal{L}(T)$.

Beweis: Wir beweisen zuerst die folgende Behauptung:

$$\forall T \in [D_\ell, D_h] : V(C(G_{k,l}, v)) \subseteq \mathcal{L}(T, v).$$

Wir führen diesen Beweis durch Widerspruch. Sei dazu $x \in V(C(G_{k,\ell}, v)) \setminus \mathcal{L}(T, v)$. Da $x \in V(C(G_{k,\ell}, v))$ gibt es einen Pfad p von v nach x in $G_{k\ell}$ (siehe dazu auch Abbildung 7.43). Sei $(y, z) \in p$ die erste Kante des Pfades von v nach x in $G_{k\ell}$, so dass $y \in \mathcal{L}(t, v)$, aber $z \notin \mathcal{L}(T, v)$ gilt. Da $(y, z) \in E(C(G_{k,\ell}, v)) \subseteq E_{k,\ell}$ ist, gilt $D_h(y, z) < D_\ell(k, \ell)$.

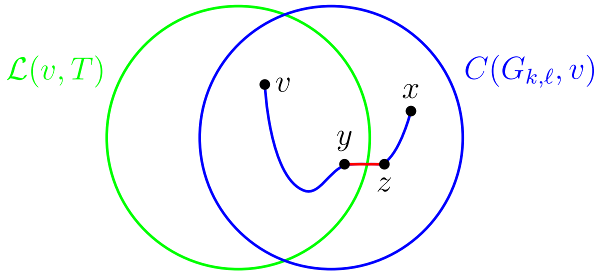


Abbildung 7.43: Skizze: $\mathcal{L}(T,v)$ und $C(G_{k,\ell},v)$

Da $y \in \mathcal{L}(T,v)$, aber $z \notin \mathcal{L}(T,v)$ ist, gilt $\mathrm{lca}(y,z) = r(T)$. Somit ist

$$d_T(y,z) \geq \mathrm{MAX}(D_\ell) = D_\ell(k,\ell).$$

Daraus folgt unmittelbar, dass

$$d_T(y,z) \geq D_\ell(k,\ell) > D_h(y,z).$$

Dies ist aber offensichtlich ein Widerspruch zu $T \in [D_\ell, D_h]$ und somit ist die Behauptung gezeigt.

Wir zeigen jetzt, wie ein Baum T umgebaut werden kann, so dass er die gewünschte Eigenschaft des Lemmas erfüllt. Sei also T ein niedriger ultrametrischer Baum mit $T \in [D_\ell, D_h]$. Sei weiter $S := \mathcal{L}(T,v) \setminus V(C(G_{k,\ell},v))$. Ist S leer, so ist nichts mehr zu zeigen. Sei also $s \in S$ und $x \in V(C(G_{k,\ell},v))$. Nach Wahl von s und x gibt es in $G_{k,\ell}$ keinen Pfad von s nach x. Somit gilt

$$D_h(x,s) \geq D_\ell(k,\ell) = \mathrm{MAX}(D_\ell) = \mathrm{MAX}(T) \geq d_T(x,s) \geq D_\ell(x,s).$$

Wir bauen jetzt T zu T' wie folgt um. Betrachte den Teilbaum T_1 der Wurzel $r(T)$ von T der sowohl x als auch s enthält. Wir duplizieren T_1 zu T_2 und hängen an die Wurzel von T'' sowohl T_1 als auch T_2 an. In T_1 entfernen wir alle Blätter aus S und in T_2 entfernen wir alle Blätter aus $V(C(G_{k,\ell},v))$ (siehe auch Abbildung 7.44). Anschließend räumen wir in den Bäumen noch auf, indem wir Kanten entfernen die zu keinen echten Blättern mehr führen. Ebenso löschen wir Knoten, die nur noch ein Kind besitzen, indem wir dieses Kind zum Kind des Elters des gelöschten Knoten machen. Letztendlich erhalten wir einen neuen ultrametrischen Baum T''.

Wir zeigen jetzt, dass $T'' \in [D_\ell, D_h]$ ist. Da wir die Knotenmarkierung der überlebenden Knoten nicht ändern, bleibt der ultrametrische Baum niedrig. Betrachten wir zwei Blätter x und y , die nicht zu T_1 oder T_2 gehören. Da der Pfad in T'' derselbe

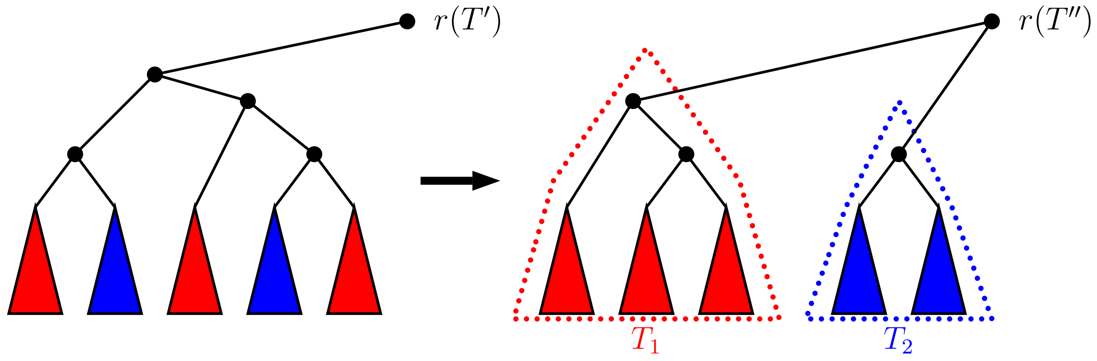


Abbildung 7.44: Skizze: Umbau des niedrigen ultrametrischen Baumes

wir T' ist, gilt weiterhin

$$D_\ell(x, y) \leq d_{T''}(x, y) \leq D_h(x, y).$$

Gehört ein Knoten x zu T_1 oder T_2 und der andere Knoten y nicht zu T_1 und T_2, so ist der Pfad nach die Duplikation bezüglich des Abstands derselbe. Es gilt also wiederum

$$D_\ell(x, y) \leq d_{T''}(x, y) \leq D_h(x, y).$$

Gehören beide Knoten v und w entweder zu T_1 oder zu T_2, dann hat sich der Pfad nach der Duplikation bzgl. des Abstands auch wieder nicht geändert, also gilt

$$D_\ell(x, y) \leq d_{T''}(x, y) \leq D_h(x, y).$$

Es bleibt der Fall zu betrachten, dass ein Knoten zu T_1 und einer zu T_2 gehört. Hier hat sich der Pfad definitiv geändert, da er jetzt über die Wurzel von T'' führt. Sei s der Knoten in T_1 und x der Knoten in T_2. Wir haben aber bereits gezeigt, dass für solche Knoten gilt:

$$D_h(x, s) \geq d_{T''}(x, s) \geq D_\ell(x, s).$$

Somit ist der Satz bewiesen. ■

Aus diesem Beweis ergibt sich unmittelbar die folgende Idee für unseren Algorithmus. Aus der Kenntnis des Graphen $G_{k\ell}$ für ein größte untere Schranke $D_\ell(k, \ell)$ für zwei Spezies k und ℓ beschreiben uns die Zusammenhangskomponenten die Partition der Spezies, wie diese in den verschiedenen Teilbäumen an der Wurzel des ultrametrischen Baumes hängen müssen, sofern es überhaupt einen gibt. Somit können wir die Spezies partitionieren und für jede Menge der Partition einen eigenen ultrametrischen Baum rekursiv konstruieren, deren Wurzeln dann die Kinder der Gesamtwurzel des zu konstruierenden ultrametrischen Teilbaumes werden. Damit ergibt sich der folgende, in Abbildung 7.45 angegebene Algorithmus.

- Bestimme $k, \ell \in [1:n]$, so dass $D_\ell(k,\ell) = \text{MAX}(D_\ell)$. $O(n^2)$
- Konstruiere $G_{k,\ell}$. $O(n^2)$
- Bestimme die Zusammenhangskomponenten $C_1, \ldots, C_m$ von $G_{k,\ell}$. $O(n^2)$
- Konstruiere rekursiv ultrametrische Bäume für die einzelnen Zusammenhangskomponenten. $\sum_{i=1}^m T(|C_i|)$
- Baue aus den Teillösungen $T_1, \ldots, T_m$ für $C_1, \ldots, C_m$ einen ultrametrischen Baum, indem man die Wurzeln der $T_1, \ldots, T_m$ als Kinder an eine neue Wurzel hängt, die als Knotenmarkierung $\text{MAX}(D_\ell)$ erhält. $O(n)$

Abbildung 7.45: Algorithmus: Algorithmus für das ultrametrische Sandwich-Problem

Für die Korrektheit müssen wir nur noch zeigen, dass die Partition nicht trivial ist, d.h., dass der Graph $G_{k,\ell}$ nicht zusammenhängend ist.

Lemma 7.44 *Seien $D_\ell \leq D_h$ zwei $n \times n$-Distanzmatrizen und seien $k, \ell \in [1:n]$ so gewählt, dass $D_\ell(k,\ell) = \text{MAX}(D_\ell)$ gilt. Wenn $G_{k,\ell}$ zusammenhängend ist, dann kann es keine ultrametrische Matrix $U \in [D_\ell, D_h]$ geben.*

Beweis: Wir führen den Beweis durch Widerspruch. Angenommen $G_{k,\ell}$ ist zusammenhängend und es existiert eine ultrametrische Matrix U mit zugehörigem Baum T. Sei $D_\ell(k,\ell) = \text{MAX}(D_\ell)$. Wir wissen, dass $U(k,\ell) \geq D_\ell(k,\ell)$, weil $U \in [D_\ell, D_h]$. Weiterhin ist der kleinste gemeinsame Vorfahre $v = \text{lca}(k,\ell)$ von k und ℓ mit $U(k,\ell)$ markiert.

Sei r_k bzw. r_ℓ das Kind von v, deren Teilbaum k bzw. ℓ enthält. Für alle Blätter i, j aus den Teilbäumen T_{r_k}, T_{r_ℓ} mit $i \in T_{r_k}$ und $j \in T_{r_\ell}$ gilt: $U(i,j) = U(k,\ell)$. Außerdem sind die Blattmengen $\mathcal{L}(T_{r_k})$ und$\mathcal{L}(T_{r_\ell})$ disjunkt.

Seien C_k die von k in $G_{k,\ell}$ erreichbaren Knoten. Aus der Definition folgt, dass

$$\forall i, j \in C_k : \ U(i,j) \leq D_h(i,j) < \text{MAX}(D_\ell) \leq U(k,\ell).$$

Aus $U(i,k) < U(k,\ell)$ folgt, dass die Blätter i und k einen niedrigsten gemeinsamen Vorfahren haben, deren Markierung größer ist als die von v ist. Daher gilt $i \in T_{r_k}$.

Unter der Annahme, dass $G_{k,\ell}$ zusammenhängend ist, gilt aber $\ell \in C_k$. Daraus würde auch $\ell \in T_{r_k}$ und letztlich $v = r_k$. Dies liefert den gewünschten Widerspruch ■

Damit ist die Korrektheit bewiesen. In Abbildung 7.46 ist ein Beispiel zur Illustration der Vorgehensweise des einfachen Algorithmus angegeben.

Für die Laufzeit erhalten wir

$$T(n) = O(n^2) + \sum_{i=1}^{m} T(n_i)$$

mit $\sum_{i=1}^{m} n_i = n$. Wir überlegen uns, was bei einem rekursiven Aufruf geschieht. Bei jedem rekursiven Aufruf wird eine Partition der Menge $[1 : n]$ verfeinert. Da wir mit der trivialen Partition $\{[1 : n]\}$ starten und mit $\{\{1\}, \ldots, \{n\}\}$ enden, kann es also maximal $n - 1$ rekursive Aufrufe geben. Somit ist die Laufzeit durch $O(n \cdot n^2)$ beschränkt.

Theorem 7.45 *Seien $D_\ell \leq D_h$ zwei $n \times n$-Distanzmatrizen. Ein ultrametrischer Baum $T \in [D_\ell : D_h]$ kann in Zeit $O(n^3)$ konstruiert werden, sofern überhaupt einer existiert.*

7.5.3 Charakterisierung einer effizienteren Lösung

In diesem Abschnitt wollen wir zeigen, dass wir das ultrametrische Sandwich Problem sogar in Zeit $O(n^2)$ lösen können. Dies ist optimal, da ja bereits die Eingabematrizen die Größe $\Theta(n^2)$ besitzen. Dazu benötigen wir erst noch die Definition der Kosten eines Pfades.

Definition 7.46 *Sei $G = (V, E, \gamma)$ ein gewichteter Graph. Die* Kosten $c(p)$ *eines Pfades $p = (v_0, \ldots, v_n)$ ist definiert durch*

$$c(p) := \max\left\{\gamma(v_{i-1}, v_i) \ : \ i \in [1 : n]\right\}.$$

Mit $D(G, v, w)$ bezeichnen wir die minimalen Kosten eines Pfades von v nach w in G.

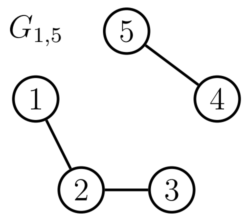
$$D(G, v, w) := \min\left\{c(p) \ : \ p \text{ ist ein Pfad von } v \text{ nach } w\right\}.$$

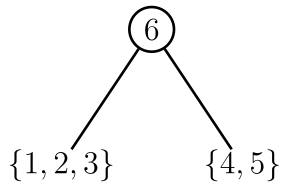
Warum interessieren wir uns überhaupt für die Kosten eines Pfades? Betrachten wir einen Pfad $p = (v_0, \ldots, v_n)$ in einem gewichteten Graphen $G(D)$, der von einer ultrametrischen Matrix D induziert wird. Seien dabei $\gamma(v, w)$ die Kosten der Kante (v, w). Wie groß ist nun der Abstand $d_D(v_0, v_n)$ von v_0 zu v_n? Unter Berücksichtigung der ultrametrischen Dreiecksungleichung erhalten wir:

$$d_D(v_0, w_n) \quad \leq \quad \max\{d_D(v_0, v_{n-1}), \gamma(v_{n-1}, v_n)\}$$

D_ℓ	1	2	3	4	5
1	0	1	2	3	6
2		0	4	5	5
3			0	4	5
4				0	1
5					0

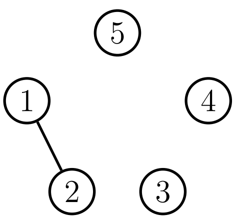
D_h	1	2	3	4	5
1	0	3	6	8	8
2		0	5	6	8
3			0	6	8
4				0	3
5					0

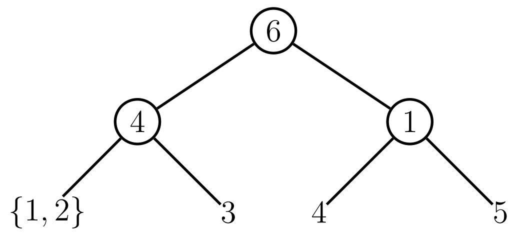




D_ℓ	1	2	3	4	5
1	0	1	2	3	6
2		0	4	5	5
3			0	4	5
4				0	1
5					0

D_h	1	2	3	4	5
1	0	3	6	8	8
2		0	5	6	8
3			0	6	8
4				0	3
5					0





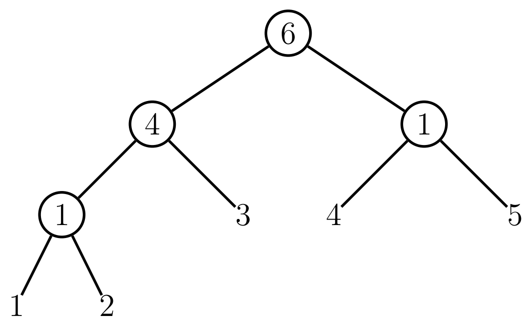


Abbildung 7.46: Beispiel: Einfache Lösung des ultrametrischen Sandwich-Problems

$$\begin{aligned}
&\leq \max\{\max\{d_D(v_0, v_{n-2}), \gamma(v_{n-2}, v_{n-1})\}, \gamma(v_{n-1}, v_n)\}\\
&= \max\{d_D(v_0, v_{n-2}), \gamma(v_{n-2}, v_{n-1}), \gamma(v_{n-1}, v_n)\}\\
&\vdots\\
&\leq \max\{\gamma(v_0, v_1), \ldots, \gamma(v_{n-2}, v_{n-1}), \gamma(v_{n-1}, v_n)\}\\
&= c(p)
\end{aligned}$$

Die letzte Gleichung folgt nur für den Fall, dass wir einen Pfad mit minimalen Kosten gewählt haben. Somit sind die Kosten eines Pfades in der zur ultrametrischen Matrix gehörigem gewichteten Graphen eine obere Schranke für den Abstand der Endpunkte dieses Pfades.

Im folgenden Lemma erhalten wir dann eine weitere Charakterisierung, wann zwei Knoten k und ℓ im zugehörigen Graphen $G_{k\ell}$ durch einen Pfad verbunden sind. Man beachte, dass wir hier nicht beliebige Knotenpaare betrachten, sondern genau das Knotenpaar, dessen maximaler Abstand gemäß der unteren Schranke den Graphen $G_{k\ell}$ definiert.

Lemma 7.47 *Seien $D_\ell \leq D_h$ zwei Distanzmatrizen. Zwei Knoten k und ℓ befinden sich genau dann in derselben Zusammenhangskomponente von $G_{k,\ell}$, wenn $D_\ell(k, \ell) > D(G(D_h), k, \ell)$.*

Beweis: **⇒:** Wenn sich k und ℓ in derselben Zusammenhangskomponente von $G_{k,\ell}$ befinden, dann gibt es einen Pfad p von k nach ℓ. Für alle Kanten $(v, w) \in p$ gilt daher (da sie Kanten in $G_{k,\ell}$ sind): $\gamma(v, w) < D_\ell(k, l)$. Somit ist auch das Maximum der Kantengewichte durch $D_\ell(k, \ell)$ beschränkt und es gilt $D(G(D_h), k, \ell) < D_\ell(k, \ell)$.

⇐: Gelte nun $D(G(D_h), k, \ell) < D_\ell(k, \ell)$. Dann existiert nach Definition von $D(\cdot, \cdot)$ und $G_{k,\ell}$ ein Pfad p in $G(D_h)$, so dass das Gewicht jeder Kante durch $D_\ell(k, \ell)$ beschränkt ist. Somit ist p auch ein Pfad in $G_{k,\ell}$ von v nach w. Also befinden sich v und w in derselben Zusammenhangskomponente von $G_{k,\ell}$. ■

Notation 7.48 *Sei T ein Baum. Den eindeutigen einfachen Pfad von $v \in V(T)$ nach $w \in V(T)$ bezeichnen wir mit $p_T(v, w)$.*

Im Folgenden sei T ein minimaler Spannbaum von $G(D_h)$. Mit obiger Notation gilt dann, dass $D(T, v, w) = c(p_T(v, w))$. Wir werden jetzt zeigen, dass wir die oberen Schranken, die eigentlich durch die Matrix D_h gegeben sind, durch den zugehörigen minimalen Spannbaum von $G(D_h)$ mit viel weniger Speicherplatz darstellen können.

Lemma 7.49 *Sei D_h eine Distanzmatrix und sei T ein minimaler Spannbaum des Graphen $G(D_h)$. Dann gilt $D(T, v, w) = D(G(D_h), v, w)$ für alle Knoten $v, w \in V(T) = V(G(D_h))$.*

Beweis: Zuerst halten wir fest, dass jeder Pfad in T auch ein Pfad in $G(D_h)$ ist. Somit gilt in jedem Falle

$$D(G(D_h), v, w) \leq D(T, v, w).$$

Für einen Widerspruchsbeweis nehmen wir jetzt an, dass es zwei Knoten v und w mit

$$D(G(D_h), v, w) < D(T, v, w)$$

gibt. Dann existiert ein Pfad p in $G(D_h)$ von v nach w mit $c(p) < D(T, v, w)$.

Wir betrachten jetzt den eindeutigen Pfad $p_T(v, w)$ im minimalen Spannbaum T. Sei jetzt (x, y) eine Kante in $p_T(v, w)$mit maximalem Gewicht, also $\gamma(x, y) = c(p_T)$. Wir entfernen jetzt diese Kante aus T und erhalten somit zwei Teilbäume T_1 und T_2, die alle Knoten des Graphen $G(D_h)$ beinhalten. Dies ist in der Abbildung 7.47 illustriert.

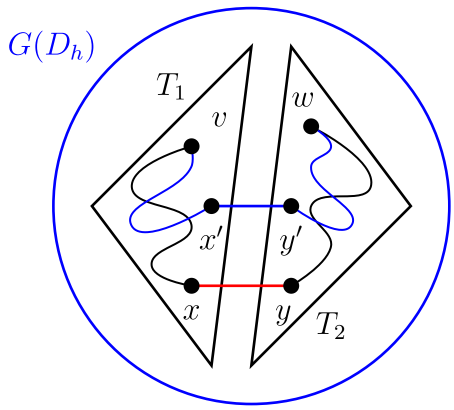


Abbildung 7.47: Skizze: Spannender Wald $\{T_1, T_2\}$ von $G(D_h)$ nach Entfernen von $\{x, y\}$ aus dem minimalen Spannbaum T

Sei (x', y') die erste Kante auf dem Pfad p in $G(D_h)$, die die Bäume T_1 und T_2 verbindet, d.h. $x' \in V(T_1)$ und $y' \in V(T_2)$. Nach Voraussetzung gilt, dass

$$D_h(x', y') < D_h(x, y),$$

da jede Kante des Pfades p nach Widerspruchsannahme leichter sein muss als die schwerste Kante in p_T und da (x, y) ja eine schwerste Kante in p_T war.

Dann könnten wir jedoch einen neuen Spannbaum T' mittels

$$E(T') = (E(T) \setminus \{(x,y)\}) \cup \{(x',y')\}$$

konstruieren. Dieser hat dann Gewicht

$$\gamma(T') \; = \; \gamma(T) + \underbrace{\gamma(x',y' - \gamma(x,y)}_{<0} \; < \; \gamma(T).$$

Somit hätten wir einen Spannbaum mit einem kleineren Gewicht konstruiert als der des minimalen Spannbaumes, was offensichtlich ein Widerspruch ist. ∎

Damit haben wir gezeigt, dass wir im Folgenden die Informationen der Matrix D_h durch die Informationen im minimalen Spannbaum T von $G(D_h)$ ersetzen können.

Definition 7.50 *Seien $D_\ell \leq D_h$ zwei $n \times n$-Distanzmatrizen. Zwei Knoten $v, w \in [1:n]$ heißen* separabel, *wenn $D_\ell(v,w) \leq D(G(D_h), v, w)$ gilt.*

Die Separabilität von zwei Knoten ist eine nahe liegende Eigenschaft, die wir benötigen werden, um zu zeigen, dass es möglich ist einen ultrametrischen Baum zu konstruieren, in dem der verbindende Pfad sowohl die untere als auch die obere Schranke für den Abstand einhält. Wir werden dies gleich formal beweisen, aber wir benötigen dazu erst noch ein paar Definitionen und Lemmata. Zuerst halten wir aber noch das folgenden Korollar fest, das aus dem vorherigen Lemma und der Definition unmittelbar folgt.

Korollar 7.51 *Seien $D_\ell \leq D_h$ zwei $n \times n$-Distanzmatrizen und sei T ein minimaler Spannbaum von $G(D_h)$. Zwei Knoten $v, w \in [1:n]$ sind genau dann separabel, wenn $D_\ell(v,w) \leq D(T,v,w)$ gilt.*

Bevor wir den zentralen Zusammenhang zwischen paarweiser Separabilität und der Existenz ultrametrischer Sandwich-Bäume zeigen, benötigen wir noch die folgende Definition.

Definition 7.52 *Seien $D_\ell \leq D_h$ zwei $n \times n$-Distanzmatrizen und sei T ein minimaler Spannbaum von $G(D_h)$. Eine Kante (x, y) des Pfades $p_T(v, w)$ im minimalen Spannbaum, der v und w verbindet, heißt* link-edge*, wenn sie eine Kante maximalen Gewichtes in $p_T(v, w)$ ist, d.h. wenn*

$$D_h(x, y) = c(p_T(v, w)) = D(T, v, w)$$

gilt. Mit $\mathrm{Link}(v, w)$ *bezeichnen wir die Menge der Link-edges für das Knotenpaar (v, w).*

Anschaulich ist die Link-Edge eines Pfades im zur oberen Schrankenmatrix gehörigen Graphen diejenige, die den maximalen Abstand von zwei Knoten bestimmt, die durch diesen Pfad verbunden werden. Aus diesem Grund werden diese eine besondere Rolle spielen.

Definition 7.53 *Seien $D_\ell \leq D_h$ zwei $n \times n$-Distanzmatrizen und sei T ein minimaler Spannbaum von $G(D_h)$. Für jede Kante $(x, y) \in E(T)$ im minimalen Spannbaum ist die* cut-weight $\mathrm{CW}(x, y)$ *wie folgt definiert:*

$$\mathrm{CW}(x, y) := \max \left\{ D_\ell(v, w) \ : \ (x, y) \in \mathrm{Link}(x, y) \right\}.$$

Die cut-weight einer Kante ist der maximale Mindestabstand zweier Knoten, deren Verbindungspfad diese Kante als link-edge besitzt. Um ein wenig mehr Licht in die Bedeutung der cut-weight zu bringen, betrachten wir jetzt nur die maximale auftretende cut-weight eines minimalen Spannbaumes des zu den Maximalabständen gehörigen Graphen.

Für diese maximale cut-weight c^* gilt dann $c^* = \mathrm{MAX}(D_\ell)$, wie man sich leicht überlegt. Wie im einfachen Algorithmus werden wir jetzt versuchen alle schwereren Kanten in $G_{k\ell}$ (der ja ein Teilgraph von $G(D_h)$ ist) zu entfernen. Statt $G_{k\ell}$ betrachten wir jedoch den minimalen Spannbaum T von $G(D_h)$ (der ja nach Definition auch ein Teilgraph von $G(D_H)$ ist).

In $G_{k\ell}$ werden alle schwereren Kanten entfernt, damit $G_{k\ell}$ in mehrere Zusammenhangskomponenten zerfällt. Zuerst überlegt man sich, dass es auch genügen würde, so viele von den schwersten Kanten zu entfernen, bis zwei Zusammenhangskomponenten entstehen. Dies wäre jedoch algorithmisch sehr aufwendig und würde in der Regel keinen echten Zeitvorteil bedeuten. Im minimalen Spannbaum T lässt sich dies hingegen sehr leicht durch Entfernen der Kante mit der maximalen cut-weight erzielen. Im Beweis vom übernächsten Lemma werden wir dies noch genauer sehen.

Das folgende Lemma stellt noch einen fundamentalen Zusammenhang zwischen Kantengewichten in Kreisen zu additiven Matrizen gehörigen gewichteten Graphen und der Eigenschaft einer Ultrametrik dar, die wir im Folgenden benötigen, um leicht von einer additiven Matrix nachweisen zu können, dass sie bereits ultrametrisch ist.

Lemma 7.54 *Eine additive Matrix M ist genau dann ultrametrisch ist, wenn für jede Folge $(i_1, \ldots, i_k) \in \mathbb{N}^k$ paarweise verschiedener Werte mit $k \geq 3$ gilt, dass in der Folge $(M(i_1, i_2), M(i_2, i_3), \ldots, M(i_{k-1}, i_k), M(i_k, i_1))$ das Maximum mehrfach angenommen wird.*

Beweis: **$\Leftarrow$:** Sei M eine additive Matrix und es gelte für alle $k \geq 3$ und für alle Folgen $(i_1, \ldots, i_k) \subset \mathbb{N}^k$ mit paarweise verschiedenen Folgengliedern, dass

$$\max\{M(i_1, i_2), M(i_2, i_3), \ldots, M(i_{k-1}, i_k), M(i_k, i_1)\} \tag{7.1}$$

nicht eindeutig ist.

Wenn man in (7.1) $k = 3$ einsetzt, gilt insbesondere für beliebige $a, b, c \in \mathbb{N}$, dass

$$\max\{M(a, b), M(b, c), M(c, a)\}$$

nicht eindeutig ist und damit ist M also ultrametrisch.

$\Rightarrow$: Sei M nun ultrametrisch, dann ist M auch additiv. Wir müssen nur noch (7.1) zu zeigen. Sei $S = (i_1, \ldots, i_k) \in \mathbb{N}^k$ gegeben. Wir betrachten zunächst i_1, i_2 und i_3. Aus der fundamentalen Eigenschaft einer Ultrametrik wissen wir, dass das Maximum von $(M(i_1, i_2), M(i_2, i_3), M(i_3, i_1))$ nicht eindeutig ist. Wir beweisen (7.1) per Induktion:

Gilt für j, dass das Maximum von $(M(i_1, i_2), M(i_2, i_3), \ldots, M(i_{j-1}, i_j), M(i_j, i_1))$ nicht eindeutig ist, so gilt dies auch für $j + 1$. Dazu betrachten wir i_1, i_j und i_{j+1}. Weiter wissen wir, dass $\max\{M(i_1, i_j), M(i_j, i_{j+1}), M(i_{j+1}, i_1)\}$ nicht eindeutig ist, d.h. einer der Werte ist kleiner als die anderen beiden. Betrachten wir wie sich das Maximum ändert, wenn wir $M(i_j, i_1)$ herausnehmen und dafür $M(i_j, i_{j+1})$ und $M(i_{j+1}, i_1)$ dazugeben.

Fall 1: $M(i_1, i_j)$ ist der kleinste Wert. Durch das Hinzufügen von $M(i_j, i_{j+1})$ und $M(i_{j+1}, i_1)$ kann das Maximum nicht eindeutig werden, denn beide Werte sind gleich. Liegen sie unter dem alten Maximum ändert sich nichts, liegen sie darüber, bilden beide das nicht eindeutige neue Maximum. Das Entfernen von $M(i_1, i_j)$ spielt nur eine Rolle, falls $M(i_1, i_j)$ vorher ein Maximum war. In dem Fall sind aber $M(i_j, i_{j+1})$ und $M(i_{j+1}, i_1)$ das neue nicht eindeutige Maximum.

Fall 2: $M(i_{j+1}, i_1)$ ist der kleinste Wert. Dann ist $M(i_1, i_j) = M(i_j, i_{j+1})$. Das Entfernen von $M(i_1, i_j)$ wird durch das Hinzufügen von $M(i_j, i_{j+1})$ wieder ausgeglichen. $M(i_{j+1}, i_1)$ wird auch hinzugefügt, spielt aber keine Rolle.

Fall 3: $M(i_j, i_{j+1})$ ist der kleinste Wert. Dann ist $M(i_1, i_j) = M(i_1, i_{j+1})$. Dieser Fall ist analog zu Fall 2. ∎

Nun kommen wir zum Beweis unseres zentralen Lemmas, dass es genau dann einen ultrametrischen Baum für einen gegebene Sandwich-Bedingung gibt, wenn alle Knoten paarweise separabel sind. Der Bewies in die eine Richtung wird konstruktiv sein und wird uns somit einen effizienten Algorithmus an die Hand geben.

Lemma 7.55 *Seien $D_\ell \leq D_h$ zwei $n \times n$-Distanzmatrizen. Ein ultrametrischer Baum $U \in [D_\ell, D_h]$ existiert genau dann, wenn jedes Paar $v, w \in [1 : n]$ von Knoten separabel ist.*

Beweis: **⇒:** Für einen Widerspruchsbeweis nehmen wir an, dass v und w nicht separabel sind. Dann ist $D_\ell(v, w) > D_h(x, y)$ für alle $\{x, y\} \in \text{Link}(v, w)$. Für jede Kante $\{a, b\}$ in $p_T(v, w)$ gilt dann $D_h(a, b) \leq D_h(x, y)$ für alle $\{x, y\} \in \text{Link}(v, w)$. Also ist $\{v, w\} \notin p_T(v, w)$. Damit ist $D_\ell(x, y) > D_h(a, b)$ für alle $\{a, b\} \in \text{Link}(v, w)$. Damit muss die Kante $\{x, y\}$ im Kreis gebildet aus $p_T(x, y)$ und der Kante $\{x, y\}$ in einer ultrametrischen Matrix das eindeutige Maximum sein. Dies steht jedoch im Widerspruch zu Lemma 7.54 und diese Implikation ist bewiesen.

⇐: Sei T ein minimaler Spannbaum von $G(D_h)$ und sei $E(T) = \{e_1, \ldots, e_{n-1}\}$, wobei $\text{CW}(e_1) \geq \cdots \geq \text{CW}(e_{n-1})$. Wir entfernen jetzt sukzessive die Kanten aus T gemäß der cut-weight der Kanten. Dabei wird durch jedes Entfernen ein Baum in zwei neue Bäume zerlegt.

Betrachten wir jetzt nachdem Entfernen der Kanten $e_1, \ldots, e_{i-1}$ den entstandenen Wald und darin den Baum T', der die Kante $e_i = \{v, w\}$ enthält. Das Entfernen der Kante $e_i = \{v, w\}$ zerlegt den Baum T' in zwei Bäume T'_1 und T'_2. Wir setzten dann $d_U(v, w) := \text{CW}(e_i)$ für alle $v \in V(T'_1)$ und $w \in V(T'_2)$.

Wir zeigen als erstes, dass $d_U(v, w) \geq D_\ell(v, w)$ für alle $v, w \in [1 : n]$. Wir betrachten die Kante e, nach deren Entfernen die Knoten v und w im Rest des minimalen Spannbaums nicht mehr durch einen Pfad verbunden sind. Ist e eine link-edge in $p_T(v, w)$, dann gilt nach Definition der cut-weight: $\text{CW}(e) \geq D_\ell(v, w)$. Ist e hingegen keine link-edge in $p_T(v, w)$, dann gilt $\text{CW}(e) \geq \text{CW}(e')$ für jede link-edge $e' \in \text{Link}(v, w)$, da wir die Kanten gemäß ihrer absteigenden cut-weight aus dem minimalen Spannbaum T entfernen. Somit gilt nach Definition der cut-weight:

$$\mathrm{CW}(e) \geq \mathrm{CW}(e') \geq D_\ell(v, w).$$

Wir zeigen jetzt, dass ebenfalls $d_U(v, w) \leq D_h(v, w)$ für alle $v, w \in [1 : n]$ gilt. Nach Definition der cut-weight gilt, dass ein Knotenpaar $\{x, y\}$ existiert, so dass für alle $\{a, b\} \in \mathrm{Link}(x, y)$ gilt: $D_\ell(x, y) = \mathrm{CW}(a, b)$. Da x und y nach Voraussetzung separabel sind, gilt

$$\mathrm{CW}(a, b) = D_\ell(x, y) \leq D(T, x, y) = D_h(a, b).$$

Die letzte Gleichheit folgt aus der Tatsache, dass $\{a, b\} \in \mathrm{Link}(x, y)$. Aufgrund der Konstruktion des minimalen Spannbaumes gilt weiterhin $D_h(a, b) \leq D_h(v, w)$. Somit gilt $d_U(v, w) = \mathrm{CW}(a, b) \leq D_h(v, w)$.

Damit haben wir gezeigt, dass $U \in [D_\ell, D_h]$. Wir müssen zum Schluss nur noch zeigen, dass U ultrametrisch ist. Nach Lemma 7.54 genügt es zu zeigen, dass in jedem Kreis in $G(U)$ das Maximum nicht eindeutig ist. Sei also C ein Kreis in $G(U)$ und (v, w) eine Kante maximalen Gewichtes in C. Falls es mehrere davon geben sollte, wählen wir eine solche, deren Endpunkte in unserem Konstruktionsverfahren durch Entfernen von Kanten im minimalen Spannbaum T zuerst separiert wurden.

Wir betrachten jetzt den Zeitpunkt in unserem Konstruktionsverfahren von d_U, als $d_U(v, w)$ gesetzt wurde. Zu diesem Zeitpunkt wurde v und w in zwei Bäume T' und T'' aufgeteilt. Ferner sind die Knoten dieses Kreises nach Wahl der Kante $\{v, w\}$ alle Knoten des Kreises in diesen beiden Teilbäumen enthalten. Da es in C noch einen anderen Weg von v nach w gibt, muss zu diesem Zeitpunkt auch für eine andere Kante $\{v', w'\}$ der Wert $d_U(v', w')$ ebenfalls festgelegt worden sein. Nach unserer Konstruktion gilt dann natürlich $d_U(v, w) = d_U(v', w')$ und in C ist das Maximum nicht eindeutig. ■

7.5.4 Algorithmus für das ultrametrische Sandwich-Problem

Der Beweis des vorherigen Lemmas liefert unmittelbar den folgenden Algorithmus, der in Abbildung 7.48 angegeben ist. Die Korrektheit des Algorithmus folgt im Wesentlichen aus dem Beweis des vorherigen Lemmas (wir gehen später noch auf ein paar implementierungstechnische Besonderheiten ein). Wir müssen uns nur noch um die effiziente Implementierung kümmern. Einen Algorithmus zur Bestimmung minimaler Spannbäume haben wir bereits kennen gelernt. Wir werden hier noch eine andere Möglichkeit darstellen, die für unsere Zwecke besser geeignet ist. Ebenso müssen wir uns noch um die effiziente Berechnung der cut-weights kümmern. Außerdem müssen wir noch erklären was kartesische Bäume sind und wie wir sie konstruieren können.

1. Bestimme minimalen Spannbaum T für $G(D_h)$. $O(n^2)$
2. Bestimme dabei den Kartesischen Baum R für den Zusammenbau von T $O(n^2)$
3. Bestimme den cut-weight der einzelnen Kanten aus T mit Hilfe des Kartesischen Baumes. $O(n^2)$
4. Baue den minimalen Spannbaum durch Entfernen der Kanten absteigend nach den cut-weights der Kanten ab und bauen parallel den ultrametrischen Baum U wieder auf. $O(n)$

Abbildung 7.48: Algorithmus: Effiziente Lösung des ultrametrische Sandwich-Problems

7.5.4.1 Kruskals Algorithmus für minimale Spannbäume

Zuerst stellen wir eine alternative Methode zu Prims Algorithmus zur Berechnung minimaler Spannbäume vor. Auch hier werden wir wieder den Greedy-Ansatz verwenden. Wir beginnen jedoch anstatt mit einem Baum aus einem Knoten, den wir sukzessive zu einem Spannbaum erweitern, mit einem Wald von Bäumen, die zu Beginn aus einelementigen Bäumen bestehen. Dabei stellt jeder Knoten genau einen Baum dar. Dann fügen wir sukzessive Kanten in diesem Wald hinzu, um dabei zwei Bäume mittels dieser Kante zu einem neuen größeren Baum zu verschmelzen. Nachdem wir $n-1$ Kanten hinzugefügt haben, haben wir unseren Spannbaum konstruiert.

Da wir gierig vorgehen, d.h. leichte Kanten bevorzugen, werden wir zuerst die Kanten aufsteigend nach Gewicht sortieren und versuchen diese in dieser Reihenfolge zu verwenden. Dabei müssen wir nur beachten, dass wir keine Kreise generieren (da Bäume durch Kreisfreiheit und Zusammenhang charakterisiert sind). Dazu merken wir uns mit einer so genannten *Union-Find-Datenstruktur* wie die Mengen der Knoten auf die verschiedenen Menge im Wald verteilt sind. Wenn wir feststellen, dass eine Kante innerhalb eines Baumes (also innerhalb einer Menge) verlaufen würde, dann verwerfen wir sie, andernfalls wird sie aufgenommen. Dieser Algorithmus ist im Detail in Abbildung 7.49 angegeben.

Halten wir noch, dass Ergebnis fest, wobei wir im nächsten Teilabschnitt noch zeigen werden, dass $\mathcal{UF}(n) = O(n^2)$ ist.

Lemma 7.56 *Sei $G = (V, E, \gamma)$ ein gewichteter Graph. Ein minimaler Spannbaum T für G kann mit Hilfe des Algorithmus von Kruskal in Zeit $O(n^2 + \mathcal{UF}(n))$ berechnet werden, wobei $\mathcal{UF}(n)$ die Zeit ist, die eine Union-Find-Datenstruktur bei n^2 Find- und n Union-Operationen benötigt.*

```
KRUSKAL (G = (V, E, γ))
{
    /* O.B.d.A. gelte γ(e_1) ≤ ··· ≤ γ(e_m) mit E = {e_1, ..., e_m} */
    set E' = ∅;
    /* (V, E') wird der konstruierte minimale Spannbaum sein */
    for (i = 1; i ≤ m; i++)
    {
        Sei e_i = {v, w};
        k = FIND(v);
        ℓ = FIND(w);
        if (k ≠ ℓ)
        {
            E' = E' ∪ {e_i};
            UNION(k, ℓ);
            if (|E'| = |V| − 1) return (V, E');
        }
    }
}
```

Abbildung 7.49: Algorithmus: Kruskals Algorithmus für minimale Spannbäume

7.5.4.2 Union-Find-Datenstruktur

Jetzt müssen wir noch genauer auf die so genannte Union-Find-Datenstruktur eingehen. Eine Union-Find-Datenstruktur für eine Grundmenge U beschreibt eine Partition $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ für U mit $U = \bigcup_{i=1}^{\ell} P_i$ und $P_i \cap P_j = \emptyset$ für alle $i \neq j \in [1 : \ell]$. Dabei ist zu Beginn $\mathcal{P} = \{P_1, \ldots, P_{|U|}\}$ mit $P_u = \{u\}$ für alle $u \in U$. Weiterhin werden die beiden folgenden elementaren Operationen zur Verfügung gestellt:

Find-Operation: Gibt für eine Element $u \in U$ den Index der Menge in der Mengenpartition zurück, die u enthält.

Union-Operation: Vereinigt die beiden Menge mit Index i und Index j und vergibt für diese vereinigte Menge einen neuen Index.

Die Realisierung erfolgt durch zwei Felder. Dabei nehmen wir der Einfachheit halber an, dass $U = [1 : n]$ ist. Ein Feld von ganzen Zahlen namens Index gibt für jedes Element $u \in U$ an, welchen Index die Menge besitzt, die u enthält. Ein weiteres Feld von Listen namens Liste enthält für jeden Mengenindex eine Liste von Elementen, die in der entsprechenden Menge enthalten sind.

Die Implementierung der Prozedur Find ist nahe liegend. Es wird einfach der Index zurück gegeben, der im Feld Index gespeichert ist. Für die Union-Operation wer-

```
UNION-FIND (int n)
function INITIALIZE(int n)
{
    int Index[n];
    for (i = 1; i ≤ n; i++)
        Index[i] = i;
    <int> Liste[n];
    for (i = 1; i ≤ n; i++)
        Liste[i] = <i>;
}

function FIND(int u)
{
    return Index[u];
}

function UNION(int i, j)
{
    if (Liste[i].size() ≤ Liste[j].size())
    {
        for all (u ∈ Liste[i]) do
        {
            Liste[j].add(u);
            Index[u] = j;
            Liste[i].remove(u);
        }
    }
    else
    {
        for all (u ∈ Liste[j]) do
        {
            Liste[i].add(u);
            Index[u] = i;
            Liste[j].remove(u);
        }
    }
}
```

Abbildung 7.50: Algorithmus: Union-Find

den wir die Elemente einer Menge in die andere kopieren. Die umkopierte Menge wird dabei gelöscht und der entsprechende Index der wiederverwendeten Menge wird dabei recycelt. Um möglichst effizient zu sein, werden wir die Elemente der kleineren Menge in die größere Menge kopieren. Die detaillierte Implementierung ist in Abbildung 7.50 angegeben.

Wir überlegen uns jetzt noch die Laufzeit dieser Union-Find-Datenstruktur. Hierbei nehmen wir an, dass wir k Find-Operationen ausführen und maximal $n-1$ Union-Operationen. Mehr Union-Operationen machen keinen Sinn, da sich nach $n-1$ Union-Operationen alle Elemente in einer Menge befinden.

Offensichtlich kann jede Find-Operation in konstanter Zeit ausgeführt werden. Für die Union-Operation ist der Zeitbedarf proportional zur Anzahl der Elemente in der kleineren Menge, die in der Union-Operation beteiligt ist. Somit ergibt sich für die maximal $n-1$ möglichen Union-Operationen.

$$\sum_{\sigma=(L,L')} \sum_{\substack{i \in L \\ |L| \leq |L'|}} O(1).$$

Um diese Summe jetzt besser abschätzen zu können vertauschen wir die Summationsreihenfolge. Anstatt die äußere Summe über die Union-Operation zu betrachten, summieren wir für jedes Element in der Grundmenge, wie oft es bei einer Union-Operation in der kleineren Menge sein könnte.

$$\sum_{\sigma=(L,L')} \sum_{\substack{i \in L \\ |L| \leq |L'|}} O(1) = \sum_{u \in U} \sum_{\substack{\sigma=(L,L') \\ u \in L \\ |L| \leq |L'|}} O(1).$$

Was passiert mit einem Element, dass sich bei einer Union-Operation in der kleineren Menge befindet? Danach befindet es sich in einer Menge die mindestens doppelt so groß wie vorher ist, da diese mindestens so viele neue Element in die Menge hinzubekommt, wie vorher schon drin waren. Damit kann jedes Element maximal $\log(n)$ Mal in einer kleineren Menge bei einer Union-Operation gewesen sein, da sich dieses Element dann in einer Menge mit mindestens n Elementen befinden muss.

Da die Grundmenge aber nur n Elemente besitzt, kann danach überhaupt keine Union-Operation mehr ausgeführt werden, da sich dann alle Elemente in einer Menge befinden. Somit ist die Laufzeit für ein Element durch $O(\log(n))$ beschränkt. Da es maximal n Elemente gibt, ist die Gesamtlaufzeit aller Union-Operationen durch $O(n\log(n))$ beschränkt.

Theorem 7.57 *Sei U mit $|U| = n$ die Grundmenge für die vorgestellte Union-Find-Datenstruktur. Die Gesamtlaufzeit von k Find- und maximal $n-1$ Union-Operationen ist durch $O(k + n\log(n))$ beschränkt.*

Es gibt noch effizienter Implementierungen von Union-Find-Operationen, die wir hier aber nicht benötigen und daher auch nicht näher darauf eingehen wollen. Wir verweisen statt dessen auf die einschlägige Literatur.

7.5.4.3 Kartesische Bäume

Kommen wir nun zur Definition eine kartesischen Baumes.

Definition 7.58 *Sei M eine Menge von Objekten, $E \subset \binom{M}{2}$ eine Menge von Paaren, so dass der Graph (M, E) ein Baum ist und $\gamma : E \to \mathbb{R}$ eine Gewichtsfunktion auf E. Ein* kartesischer Baum *für (M, E) ist rekursiv wie folgt definiert.*

- *Ist $|M| = 1$, dann ist der einelementige Baum mit der Wurzelmarkierung $m \in M$ ein kartesischer Baum.*
- *Ist $|M| \geq 2$ und $\{v_1, v_2\} \in E$ mit $\gamma(\{v_1, v_2\}) = \max\{\gamma(e) : e \in E\}$. Sei weiter T' der Wald, der aus T durch Entfernen von $\{v_1, v_2\}$ entsteht, d.h. $V(T') = V(T)$ und $E(T') = E(T) \setminus \{v_1, v_2\}$. Sind T_1 bzw. T_2 kartesische Bäume für $C(T', v_1)$ bzw. $C(T', v_2)$, dann ist der Baum mit der Wurzel, die mit $\{v_1, v_2\}$ markiert ist und deren Teilbäume der Wurzel gerade T_1 und T_2 sind, ebenfalls ein kartesischer Baum.*

Wir bemerken hier noch explizit an, dass die Elemente aus M nur als Blattmarkierungen auftauchen und dass alle inneren Knoten mit den Elementen aus E markiert sind.

Betrachtet man auf der Menge $[1 : n]$ als Baum eine lineare Liste mit

$$E = \{\{i, i+1\} : i \in [1 : n-1]\} \quad \text{mit} \quad \gamma(i, i+1) = \max\{F[i], F[i+1]\},$$

wobei F ein Feld von Zahlen ist, so erhält man im Wesentlichen einen Heap, wobei hier die Werte an den Blättern stehen und die inneren Knoten den maximalen Wert ihres Teilbaumes besitzen, wenn man, wie hier üblich, anstatt der Kante in den inneren Knoten das Gewicht der Kante einträgt. Diese Struktur wird manchmal auch als kartesischer Baum bezeichnet wird.

Der kartesische Baum für einen minimalen Spannbaum lässt sich jetzt bei der Konstruktion mit Hilfe des Kruskal-Algorithmus sehr leicht mitkonstruieren. Man überlegt sich leicht, dass bei der Union-Operation, die zugehörigen kartesischen Bäume mit einer neuen Wurzel vereinigt werden, wobei die Kante in der Wurzel genau die Kante ist, die bei der Konstruktion des minimalen Spannbaumes hinzugefügt wird.

Lemma 7.59 *Sei $G = (V, E, \gamma)$ ein gewichteter Graph und T ein minimaler Spannbaum von G. Der kartesische Baum für T kann bei der Konstruktion des minimalen Spannbaumes T basierend auf Kruskals Algorithmus parallel mit derselben Zeitkomplexität mitkonstruiert werden.*

7.5.4.4 Berechnung der cut-weights

Wir wollen jetzt zeigen, dass wir die cut-weights einer Kante $e \in E$ im minimalen Spannbaum T mit Hilfe von lca-Anfragen im kartesischen Baumes T bestimmen können. Dazu gehen wir durch die Matrix D_ℓ und bestimmen für jedes Knotenpaar (i, j) den niedrigsten gemeinsamen Vorfahren $\text{lca}(i, j)$ im kartesischen Baum R.

Die dort gespeicherte Kante e ist nach Konstruktion des kartesischen Baumes eine Kante mit größtem Gewicht auf dem Pfad von i nach j im minimalen Spannbaum T, d.h. $e \in \text{Link}(i, j)$. Daher werden wir dort die cut-weight $\text{CW}(e)$ dieser Kante mittels $\text{CW}(e) = \max\{\text{CW}(e), D_\ell(i, j)\}$ aktualisieren. Wir bemerken an dieser Stelle, dass es durchaus noch andere link-edges auf dem Pfad von i nach j im minimalen Spannbaum geben kann. Daher wird die cut-weight nicht für jede Kante korrekt berechnet. Wir gehen auf diesen „Fehler“ am Ende diese Abschnittes noch ein.

Lemma 7.60 *Sei $G = (V, E, \gamma)$ ein gewichteter Graph und T ein minimaler Spannbaum von G. Der kartesische Baum R für den minimalen Spannbaum T kann mit derselben Zeit konstruiert werden wie der minimale Spannbaum, sofern hierfür der Algorithmus von Kruskal verwendet wird.*

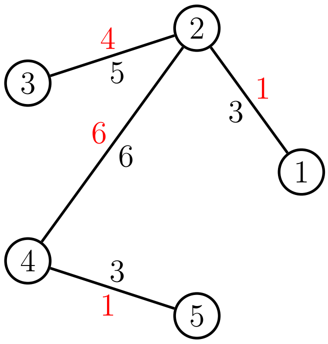
Nachdem wir jetzt die wesentlichen Schritte unseres effizienten Algorithmus weitestgehend verstanden haben, können wir uns einem Beispiel zuwenden. In der Abbildung 7.51 ist ein Beispiel angegeben, wie unser effizienter Algorithmus vorgeht.

7.5.4.5 Least Common Ancestor Queries

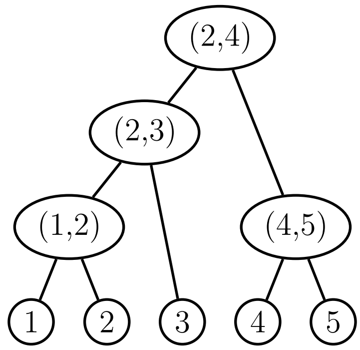
Wir müssen uns nun nur noch überlegen, wie wir $O(n^2)$ lca-Anfragen in Zeit $O(n^2)$ beantworten können. Dazu werden wir das lca-Problem auf das Range Minimum Query Problem reduzieren, das wie folgt definiert ist.

D_ℓ	1	2	3	4	5
1	0	1	2	3	6
2		0	4	5	5
3			0	4	5
4				0	1
5					0

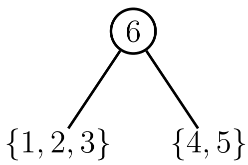
D_h	1	2	3	4	5
1	0	3	6	8	8
2		0	5	6	8
3			0	6	8
4				0	3
5					0



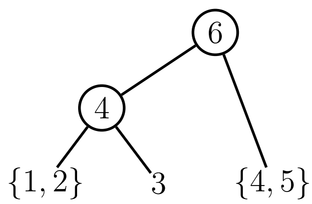
Minimaler Spannbaum T



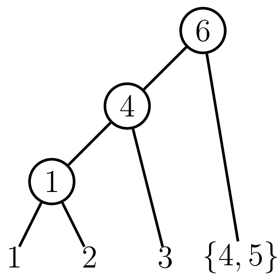
Kartesischer Baum R



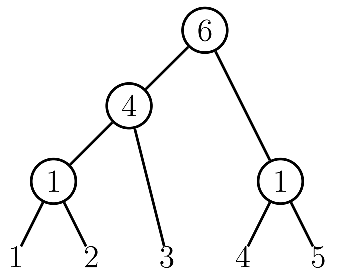
U_1 nach Entfernen $\{2, 4\}$



U_2 nach Entfernen $\{2, 3\}$



U_3 nach Entfernen $\{1, 2\}$



$U = U_4$ nach Entfernen $\{4, 5\}$

Abbildung 7.51: Beispiel: Lösung eines ultrametrischen Sandwich-Problems

RANGE MINIMUM QUERY

Eingabe: Eine Feld F der Länge n von reellen Zahlen und $i \leq j \in [1:n]$.
Ausgabe: Ein Index k mit $F[k] = \min\{F[\ell] : \ell \in [i:j]\}$.

Wir werden später zeigen, wie wir mit einem Preprocessing in Zeit $O(n^2)$ jede Anfrage in konstanter Zeit beantworten können. Für die Reduktion betrachten wir die so genannte *Euler-Tour* eines Baumes.

Definition 7.61 *Sei $T = (V, E)$ ein gewurzelter Baum mit Wurzel r und seien $T_1, \ldots, T_\ell$ die Teilbäume, die an der Wurzel hängen. Die* Euler-Tour *durch T ist eine Liste von $2n-1$ Knoten, die wie folgt rekursiv definiert ist:*

- *Ist $\ell = 0$, d.h. der Baum besteht nur aus dem Blatt r, dann ist diese Liste durch (r) gegeben.*
- *Für $\ell \geq 1$ seien $L_1, \ldots, L_\ell$ mit $L_i = (v_1^{(i)}, \ldots, v_{n_i}^{(i)})$ für $i \in [1:\ell]$ die Euler-Touren von $T_1, \ldots, T_\ell$. Die Euler-Tour von T ist dann durch*

$$(r, v_1^{(1)}, \ldots, v_{n_1}^{(1)}, r, v_1^{(2)}, \ldots, v_{n_2}^{(2)}, r, \ldots, r, v_1^{(\ell)}, \ldots, v_{n_\ell}^{(\ell)}, r)$$

  definiert.

Der Leser sei dazu aufgefordert zu verifizieren, dass die oben definierte Euler-Tour eines Baumes mit n Knoten tatsächlich Liste mit $2n-1$ Elementen ist.

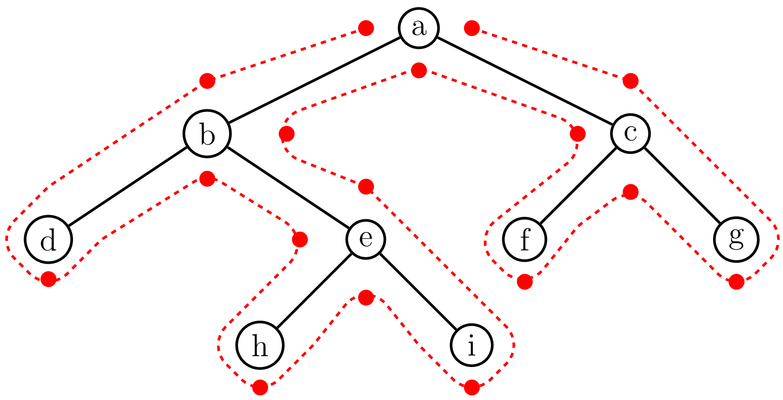
Die Euler-Tour kann sehr leicht mit Hilfe einer Tiefensuche in Zeit $O(n)$ berechnet werden. Der Algorithmus hierfür ist in Abbildung 7.52 angegeben. Man kann sich die Euler-Tour auch bildlich sehr schön als das Abmalen der Bäume anhand ihrer äußeren Kontur vorstellen, wobei bei jedem Antreffen eines Knotens des Baumes dieser in die Liste aufgenommen wird. Die ist in Abbildung 7.53 anhand eines Beispiels illustriert.

Zusammen mit der Euler-Tour, der Liste der abgelaufenen Knoten, betrachten wir zusätzlich noch DFS-Nummern des entsprechenden Knotens, die bei der Tiefensuche in der Regel mitberechnet werden (siehe auch das Beispiel in der Abbildung 7.53).

Betrachten wir jetzt die Anfrage an zwei Knoten des Baumes i und j. Zuerst bemerken wir, dass diese Knoten, sofern sie keine Blätter sind, mehrfach vorkommen können. Wir wählen jetzt für i und j willkürlich einen der Knoten, der in der Euler-Tour auftritt, als einen Repräsentanten aus. Zuerst stellen wir fest, dass in der Euler-Tour der niedrigste gemeinsame Vorfahren von i und j in der Teilliste, die durch die beiden Repräsentanten definiert ist, vorkommen muss, was man wie folgt sieht.

```
Euler-Tour (tree T = (V, E))
{
    /* r(T) bezeichne die Wurzel von T */
    output r(T);
    /* N(r(T)) bezeichne die Menge der Kinder der Wurzel von T */
    for all (v ∈ N(r(T)))
    {
        /* T(v) bezeichne den Teilbaum mit Wurzel v */
        Euler-Tour(T(v))
        output r(T);
    }
}
```

Abbildung 7.52: Algorithmus: Konstruktion einer Euler-Tour



Euler-Tour	a	b	d	b	e	h	e	i	e	b	a	c	f	c	g	c	a
DFS-Nummer	1	2	3	2	4	5	4	6	4	2	1	7	8	7	9	7	1

Abbildung 7.53: Beispiel: Euler-Tour

Wir nehmen an, dass i in der Euler-Tour vor j auftritt. In der Euler-Tour sind alle Knoten v, die nach einem Repräsentanten von i auftauchen und eine kleinere DFS-Nummer als i besitzen, ein Vorfahre von i. Da die DFS-Nummer von v kleiner als die von i ist und v in der Euler-Tour nach i auftritt, ist die DFS-Prozedur von v noch aktiv, als der Knoten i besucht wird. Das ist genau die Definition dafür, dass i ein Nachfahre von v ist. Analoges gilt für den Knoten j.

Wir betrachten jetzt nur die Knoten der Teilliste zwischen den beiden Repräsentanten von i und j, deren DFS-Nummer kleiner als die von i ist. Nach dem obigen sind dies Vorfahren von i. Nur einer davon, nämlich der mit der kleinsten DFS-Nummer, ist auch ein Vorfahre von j und muss daher der niedrigste gemeinsame Vorfahre von i und j sein. Dieser Knoten haben wir nämlich besucht, bevor wir in den Teilbaum von j eingedrungen sind. Alle Vorfahren davon haben wir mit betrachten der Teilliste eliminiert, die mit einem Repräsentanten von j endet.

Damit können wir das folgende Zwischenergebnis festhalten.

Lemma 7.62 *Gibt es eine Lösung für das Range Minimum Query Problem, dass für das Preprocessing Zeit $O(p(n))$ und für eine Anfrage $O(q(n))$ benötigt, so kann da Problem des niedrigsten gemeinsamen Vorfahren mit einen Zeitbedarf für das Preprocessing in Zeit $O(n+p(2n-1))$ und für eine Anfrage in Zeit $O(q(2n-1))$ gelöst werden.*

7.5.4.6 Range Minimum Queries

Damit können wir uns jetzt ganz auf das Range Minimum Query Problem konzentrieren. Offensichtlich kann ohne ein Preprocessing eine einzelne Anfrage mit $O(j-i) = O(n)$ Vergleichen beantwortet werden. Das Problem der Range Minimum Queries ist jedoch insbesondere dann interessant, wenn für ein gegebenes Feld eine Vielzahl von Range Minimum Queries durchgeführt werden. In diesem Fall kann mit Hilfe einer Vorverarbeitung die Kosten der einzelnen Queries gesenkt werden.

Ein triviale Lösung würde alle möglichen Anfragen vorab berechnen. Dazu könnte eine zweidimensionale Tabelle $Q[i,j]$ angelegt werden. Dazu würde für jedes Paar (i,j) das Minimum der Bereichs $F[i:j]$ mit $j-i$ Vergleiche bestimmt werden. Dies würde zu einer Laufzeit für die Vorverarbeitung von

$$\sum_{i=1}^{n}\sum_{j=i}^{n}(j-i) = \Theta(n^3)$$

führen. In der Abbildung 7.54 ist ein einfacher, auf dynamischer Programmierung basierender Algorithmus angegeben, der diese Tabelle in Zeit $O(n^2)$ berechnen kann. Damit erhalten wir das folgende Resultat für das Range Minimum Query Problem.

Theorem 7.63 *Für das Range Minimum Query Problem kann mit Hilfe einer Vorverarbeitung, die mit einem Zeitbedarf von $O(n^2)$ ausgeführt werden kann, jede Anfrage in konstanter Zeit beantwortet werden.*

Es gibt bereits wesentlich effizienter Verfahren für das Range Minimum Query Problem. In unserem Zusammenhang ist dieses leicht zu erzielende Ergebnis jedoch bereits völlig ausreichend und wir verweisen für die anderen Verfahren auf die einschlägige Literatur. Wir halten das für uns wichtige Ergebnis noch fest.

```
RMQ ( int F[], int n)
{
    for (i = 1; i ≤ n; i++)
        T[i, i] = i;

    for (i = 1; i ≤ n; i++)
        for (j = 1; i + j ≤ n; j++)
        {
            if (F[T[i, i + (j − 1)]] ≤ F[i + j])
                T[i, i + j] = T[i, i + j − 1];
            else
                T[i, i + j] = i + j;
        }
}
```

Abbildung 7.54: Algorithmus: Preprocessing für Range Minimum Queries

Theorem 7.64 *Sei T ein gewurzelter Baum mit n Knoten. Nach einer Vorverarbeitung, die in Zeit $O(n^2)$ durchgeführt werden kann, kann jede Anfrage nach einem niedrigsten gemeinsamen Vorfahren zweier Knoten aus T in konstanter Zeit beantwortet werden.*

7.5.4.7 Reale Berechnung der cut-weights

Wie bereits schon angedeutet berechnet unser effizienter Algorithmus nicht wirklich die cut-weights der Kanten des minimalen Spannbaumes. Dies passiert genau dann, wenn es im minimalen Spannbäume mehrere Kanten desselben Gewichtes gibt. Dazu betrachten wir das Beispiel, das in Abbildung 7.55 angegeben ist.

D_ℓ	1	2	3
1	0	1	3
2		0	2
3			0

D_h	1	2	3
1	0	5	6
2		0	5
3			0

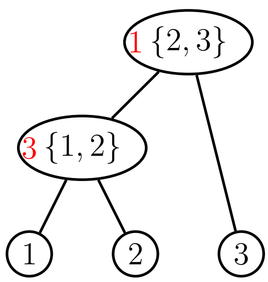


Abbildung 7.55: Beispiel: Falsche Berechnung der cut-weights

Hier stellen wir fest, das alle Kanten des minimalen Spannbaumes eine Kantengewicht von 5 besitzen. Somit sind alle Kanten des Baumes link-edges. Zuerst stellen wir fest, dass die cut-weight der Kante $\{2, 3\}$, die in der Wurzel des kartesischen Spannbaumes, mit 3 korrekt berechnet wird, da ja auch die Kante $\{1, 3\}$ der niedrigste gemeinsame Vorfahre von 1 und 3 im kartesischen Baum ist.

Im Gegensatz dazu wir die cut-weight der Kante $\{1, 2\}$ des minimalen Spannbaumes falsch berechnet. Diese Kante erhält die cut-weight 1, da die Kante $\{1, 2\}$ der niedrigste gemeinsame Vorfahre von 1 und 2 ist. Betrachten wir jedoch den Pfad von 1 über 2 nach 3, dann stellen wir fest, dass auch die Kante $\{1, 2\}$ eine link-edge im Pfad von 1 nach 3 im minimalen Spannbaum ist und die cut-weight der Kante $\{1, 2\}$ im minimalen Spannbaum daher ebenfalls 3 sein müsste.

Eine einfache Lösung wäre es die Kantengewicht infinitesimal so zu verändern, dass alle Kantengewichte im minimalen Spannbaum eindeutig wären. Wir können jedoch auch zeigen, dass der Algorithmus weiterhin korrekt ist, obwohl er die cut-weights von manchen Kanten falsch berechnet.

Zuerst überlegen wir uns, welche Kanten eine falsche cut-weight bekommen. Dies kann nur bei solchen Kanten passieren, deren Kantengewicht im minimalen Spannbaum nicht eindeutig ist. Zum anderen müssen diese Kanten bei der Konstruktion des minimalen Spannbaumes vor den anderen Kanten gleichen Gewichtes im minimalen Spannbaum eingefügt worden sein. Dies bedeutet, dass diese Kante im kartesischen Baum ein Nachfahre einer anderen Kante des minimalen Spannbaum gleichen Gewichtes sein muss.

Überlegen wir uns, was im Algorithmus passiert. Wir entfernen ja die Kanten aus dem minimalen Spannbaum nach fallendem cut-weight. Somit werden von zwei Kanten im minimalen Spannbaum, die dasselbe Kantengewicht besitzen und dieselbe cut-weight besitzen sollten, die Kante entfernt, die sich weiter oben im kartesischen Baum befindet. Damit zerfällt der minimale Spannbaum in zwei Teile und die beiden Knoten der untere Schranke die für die cut-weight der entfernten Kante verantwortlich sind befinden sich in zwei verschiedenen Zusammenhangskomponenten.

Somit ist die cut-weight der Kante, die ursprünglich falsch berechnet worden, in der neuen Zusammenhangskomponente jetzt nicht mehr ganz so falsch. Entweder ist sie für die konstruierte Zusammenhangskomponente korrekt und wir können mit unserem Algorithmus fortfahrten und er bleibt korrekt. War sie andernfalls falsch, befindet sich in minimalen Spannbaum dieser Zusammenhangskomponente eine weitere Kante mit demselben Gewicht, die im kartesischen Baum ein Vorfahre der betrachteten Kante ist und die ganze Argumentation wiederholt sich.

Also obwohl der Algorithmus nicht die richtigen cut-weights berechnet, ist zumindest immer die cut-weight der Kante mit der schwersten cut-weight korrekt berechnet und dies genügt für die Korrektheit des Algorithmus völlig, wie eine kurze Inspektion des zugehörigen Beweises ergibt.

7.5.5 Approximationsprobleme

Wir kommen jetzt noch einmal zu dem Approximationsproblem für Distanzmatrizen zurück.

ULTRAMETRISCHES APPROXIMATIONSPROBLEM

Eingabe: Eine $n \times n$-Distanzmatrizen D.
Ausgabe: Eine ultrametrische Distanzmatrix D', die $\|D - D'\|$ minimiert.

Das entsprechende additive Approximationsproblem ist leider $\mathcal{NP}$-hart. Das ultrametrische Approximationsproblem kann für die Maximumsnorm $\|\cdot\|_\infty$ in Zeit $O(n^2)$ gelöst werden. Für die anderen p-Normen ist das Problem ebenfalls wieder $\mathcal{NP}$-hart.

Theorem 7.65 *Das ultrametrische Approximationsproblem für die Maximumsnorm kann in linearer Zeit (in der Größe der Eingabe) gelöst werden.*

Beweis: Sei D die gegebene Distanzmatrix. Eigentlich müssen wir nur ein minimales $\varepsilon > 0$ bestimmen, so dass es eine ultrametrische Matrix U mit $U \in [D - \varepsilon, D + \varepsilon]$ gibt. Hierbei ist $D + x$ definiert durch $D = (d_{i,j} + x)_{i,j}$.

Wir berechnen also zuerst wieder den minimalen Spannbaum T für $G(D)$. Dann berechnen wir die cut-weights für die Kanten des minimalen Spannbaumes T. Dabei wählen wir für jede Kante e ein minimales ε_e, so dass die Knotenpaare separabel sind, d.h. $\mathrm{CW}(e) - \varepsilon_e \leq D(e) + \varepsilon_e$ für alle Kanten $e \in E(T)$.

Damit dies für alle Kanten des Spannbaumes gilt wählen ε als das Maximum dieser, d.h.

$$\varepsilon := \max\{\varepsilon_e \;:\; e \in E(T)\} = \frac{1}{2}\max\{\mathrm{CW}(e) - D(e) \;:\; e \in E(T)\}.$$

Dann führen wir denselben Algorithmus wie für das ultrametrische Sandwich Problem mit $D_\ell := D - \varepsilon$ und $D_h = D + \varepsilon$ durch. ■

Literaturhinweise

A.1 Lehrbücher zur Vorlesung

Peter Clote, Rolf Backofen: *Introduction to Computational Biology*; John Wiley and Sons, 2000.

Richard Durbin, Sean Eddy, Anders Krogh, Graeme Mitchison: *Biological Sequence Analysis*; Cambridge University Press, 1998.

Dan Gusfield: *Algorithms on Strings, Trees, and Sequences — Computer Science and Computational Biology*; Cambridge University Press, 1997.

David W. Mount: *Bioinformatics — Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press, 2001.

Pavel A. Pevzner: *Computational Molecular Biology - An Algorithmic Approach*; MIT Press, 2000.

João Carlos Setubal, João Meidanis: *Introduction to Computational Molecular Biology*; PWS Publishing Company, 1997.

Michael S. Waterman: *Introduction to Computational Biology: Maps, Sequences, and Genomes*; Chapman and Hall, 1995.

A.2 Skripten anderer Universitäten

Bonnie Berger: *Introduction to Computational Molecular Biology*, Massachusetts Institute of Technology, http://theory.lcs.mit.edu/~bab/01-18.417-home.html;

Bonnie Berger, *Topics in Computational Molecular Biology*, Massachusetts Institute of Technology, Spring 2001, http://theory.lcs.mit.edu/~bab/01-18.418-home.html;

Paul Fischer: *Einführung in die Bioinformatik* Universität Dortmund, Lehrstuhl II, WS2001/2002, http://ls2-www.cs.uni-dortmund.de/lehre/winter200102/bioinf/

Richard Karp, Larry Ruzzo: *Algorithms in Molecular Biology*; CSE 590BI, University of Washington, Winter 1998. http://www.cs.washington.edu/education/courses/590bi/98wi/

Larry Ruzzo: *Computational Biology*, CSE 527, University of Washington, Fall 2001; http://www.cs.washington.edu/education/courses/527/01au/

Georg Schnittger: *Algorithmen der Bioinformatik*, Johann Wolfgang Goethe-Universität Frankfurt am Main, Theoretische Informatik, WS 2000/2001, http://www.thi.informatik.uni-frankfurt.de/BIO/skript2.ps.

Ron Shamir: *Algorithms in Molecular Biology* Tel Aviv University, http://www.math.tau.ac.il/~rshamir/algmb.html; http://www.math.tau.ac.il/~rshamir/algmb/01/algmb01.html.

Ron Shamir: *Analysis of Gene Expression Data, DNA Chips and Gene Networks*, Tel Aviv University, 2002; http://www.math.tau.ac.il/~rshamir/ge/02/ge02.html;

Martin Tompa: *Computational Biology*, CSE 527, University of Washington, Winter 2000. http://www.cs.washington.edu/education/courses/527/00wi/

A.3 Lehrbücher zu angrenzenden Themen

Teresa K. Attwood, David J. Parry-Smith; *Introduction to Bioinformatics*; Prentice Hall, 1999.

Maxime Crochemore, Wojciech Rytter: *Text Algorithms*; Oxford University Press: New York, Oxford, 1994.

Martin C. Golumbic: *Algorithmic Graph Theory and perfect Graphs*; Academic Press, 1980.

Benjamin Lewin: *Genes*; Oxford University Press, 2000.

Milton B. Ormerod: *Struktur und Eigenschaften chemischer Verbindungen*; Verlag Chemie, 1976.

Hooman H. Rashidi, Lukas K. Bühler: *Grundriss der Bioinformatik — Anwendungen in den Biowissenschaften und der Medizin*,

Klaus Simon: *Effiziente Algorithmen für perfekte Graphen*; Teubner, 1992.

Maxine Singer, Paul Berg: *Gene und Genome*; Spektrum Akademischer Verlag, 2000.

Lubert Stryer: *Biochemie*, Spektrum Akademischer Verlag, 4. Auflage, 1996.

A.4 Originalarbeiten

Kellogg S. Booth, George S. Lueker: Testing for the Consecutive Ones property, Interval Graphs, and Graph Planarity Using PS-Tree Algorithms; *Journal of Computer and System Science*, Vol.13, 335–379, 1976.

Ting Chen, Ming-Yang Kao: On the Informational Asymmetry Between Upper and Lower Bounds for Ultrametric Evolutionary Trees, *Proceedings of the 7th Annual European Symposium on Algorithms, ESA'99*, Lecture Notes in Computer Science 1643, 248–256, Springer-Verlag, 1999.

Richard Cole: Tight Bounds on the Complexity of the Boyer-Moore String Matching Algorithm; *SIAM Journal on Computing*, Vol. 23, No. 5, 1075–1091, 1994.
s.a. *Technical Report*, Department of Computer Science, Courant Institute for Mathematical Sciences, New York University, TR1990-512, June, 1990, http://csdocs.cs.nyu.edu/Dienst/UI/2.0/Describe/ncstrl.nyu_cs%2fTR1990-512

Martin Farach, Sampath Kannan, Tandy Warnow: A Robust Model for Finding Optimal Evolutionary Trees, *Algorithmica*, Vol. 13, 155–179, 1995.

Wen-Lian Hsu: PC-Trees vs. PQ-Trees; *Proceedings of the 7th Annual International Conference on Computing and Combinatorics, COCOON 2001*, Lecture Notes in Computer Science 2108, 207–217, Springer-Verlag, 2001.

Wen-Lian Hsu: A Simple Test for the Consecutive Ones Property; *Journal of Algorithms*, Vol.43, No.1, 1–16, 2002.

Haim Kaplan, Ron Shamir: Bounded Degree Interval Sandwich Problems; *Algorithmica*, Vol. 24, 96–104, 1999.

Edward M. McCreight: A Space-Economical Suffix Tree Construction Algorithm; *Journal of the ACM*, Vol. 23, 262–272,1976.

Moritz Maaß: *Suffix Trees and Their Applications*, Ausarbeitung von der Ferienakademie '99, Kurs 2, Bäume: Algorithmik und Kombinatorik, 1999. http://www14.in.tum.de/konferenzen/Ferienakademie99/

Esko Ukkonen: On-Line Construction of Suffix Tress, *Algorithmica*, Vol. 14, 149–260, 1995.

Index

P

Q

R

S

T