Ferienakademie 1999

Kurs 2: Bäume - Algorithmik und Kombinatorik

Lowest Common Ancestors

Referent:Pham Minh Bolo

Einführung

Aufgabe. Gegeben ist ein Baum $T(V, E)$. Beantworte Anfragen der Form ”Welcher Knoten ist der kleinste gemeinsame Vorfahre (**Lowest Common Ancestor**) von zwei Knoten x, y in T ? “

Vorstellung eines Algorithmus, der nach linearer Vorverarbeitung solche LCA-Anfragen in $O(1)$ Zeit beantwortet. Parallelisierung des Verfahrens unter Verwendung einer optimalen Zahl von Prozessoren auf einer EREW PRAM liefert logarithmische Laufzeit für die Vorarbeit.

Vorarbeit

- Gegeben: Baum $T(V, E)$
- Ziel:
  - Stelle eine Tabelle HEAD auf.
  - Bestimme zu jedem Knoten v in T folgende drei Attribute:
    INLABEL(v), ASCENDANT(v), LEVEL(v).

Bestimmmung von INLABEL(v)

Schritt 1. Bestimme zu jedem Knoten v in T PREORDER(v) und SIZE(v).

Beob. PREORDER Nummern liegen in

[PREORDER(v),PREORDER(v)$+$SIZE(v)-1], dem *Intervall von* v.

Schritt 2. Wandle alle Zahlen im Intervall von v in Binärzahlen um.

Ermittle diejenige Binärzahl,die von rechts nach links gezählt

die meisten Bits ”0” bis zum ersten Auftreten einer ”1” aufweist.

Diese Zahl sei INLABEL(v).

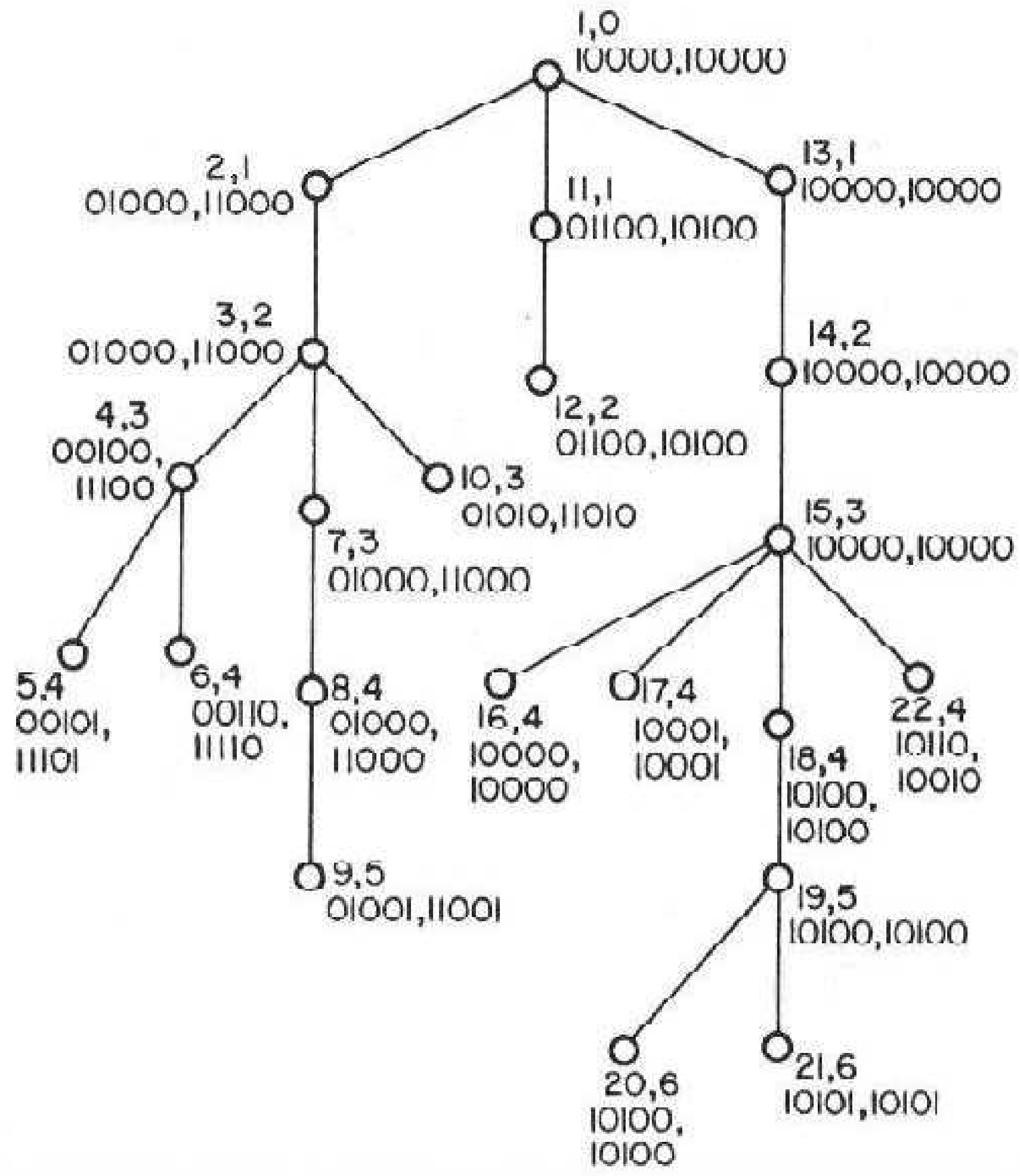

Abbildung 1. Beispiel. Vier Attribute zu jedem Knoten des Baums:

PREORDER, LEVEL, INLABEL und ASCENDANT.

Eigenschaften der INLABEL Nummern

Lemma 1. Die INLABEL Nummern erfüllen die Path-Partition Eigenschaft. Die INLABEL Nummern spalten den Baum T in Pfade auf, die sog. INLABEL Pfade. Alle einem Pfad zugehörigen Knoten besitzen die gleiche INLABEL Nummer.

Beweis Die Intervalle der Söhne von v sind paarweise disjunkt. Sei u ein Sohn von v, in dessen Intervall INLABEL(v) liegt. Es gilt dann :

INLABEL(v) $=$ INLABEL(u) (falls u existiert)

Für einen anderen Sohn w gilt : INLABEL(w) $\neq$ INLABEL(v).

$\square$

Lemma 2. Die INLABEL Nummern erfüllen die Descendance-Preservation Eigenschaft.

B sei der kleinste vollständige Binärbaum, der mindestens n Knoten besitzt. Jeder Knoten v in T wird auf einen Knoten INLABEL(v) in B abgebildet, und die Nachfolger von v in T werden auf Nachfolger von INLABEL(v) in B abgebildet werden.

Betrachte zwei beliebige Knoten b und c in B.

$l = \lfloor \log n \rfloor$. i sei die Anzahl der rechtesten Bits ”0” in b.

b besteht somit aus den $l - i$ linkesten Bits von b, gefolgt von einer ”1” und i Bits ”0”.

Behauptung: Ein Knoten c ist Nachfolger eines Knoten b genau dann, wenn

(1) die $l - i$ linkesten Bits in c gleich den $l - i$ linkesten Bits in b sind.

(2) die Anzahl der Folge von Bits ”0” in c (von rechts gezählt) höchstens i ist.

Beweis b_L und b_R seien linker und rechter Sohn von b. Es gilt:

(i) b_L ist zusammengesetzt aus den $l - i$ linkesten Bits von b, gefolgt von einer ”0”, einer ”1” und $i - 1$ Bits ”0”.

(ii) b_R ist zusammengesetzt aus den $l - i$ linkesten Bits von b, gefolgt von zwei Bits ”1” und $i - 1$ Bits ”0”.

Diese Fakten implizieren beide Richtungen der Behauptung.

$\square$

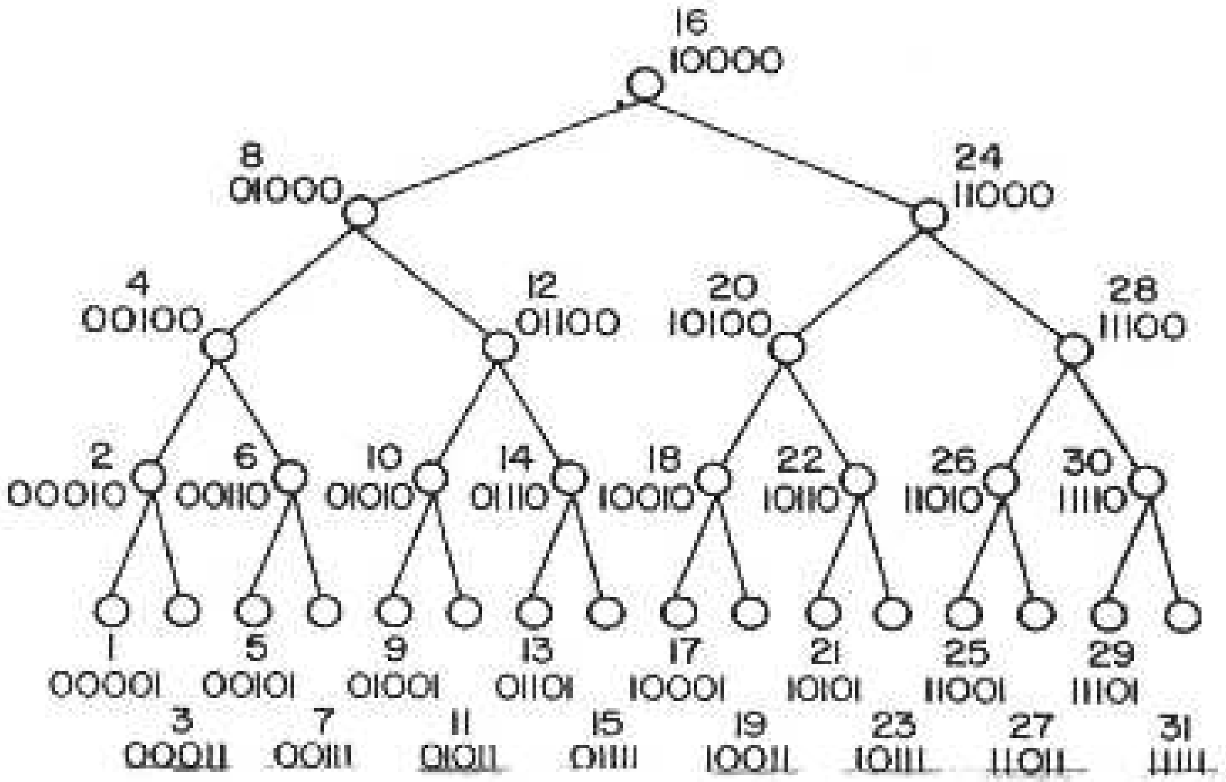

Abbildung 2. Beispiel. Inorder-Numerierung des vollständigen Binärbaums mit 31 Knoten.

Zurück zu Lemma 2. Sei d ein Nachfolger von v in T.

Behauptung: INLABEL(d) ist ein Nachfolger von INLABEL(v) in B.

Beweis Sei i die Anzahl der rechtesten Bits ”0” in INLABEL(v).

Es gilt:

(1) INLABEL(d) liegt im Intervall von v und INLABEL(v) hat die meisten Bits ”0” in diesem Intervall.

(2) Die $l - i$ linkesten Bits aller Zahlen im Intervall von v sind identisch.

Dies impliziert, dass INLABEL(d) ein Nachfolger von INLABEL(v) in B ist. Daraus folgt Lemma 2.

□

Implementierung

Implementierung von Schritt 1 in linearer Zeit und mit linearem Platzbedarf.

Falls PREORDER(v) und SIZE(v) zu jedem Knoten v in T gegeben sind, kann Schritt 2 in konstanter Zeit pro Knoten in zwei Teilschritten durchgeführt werden.

Schritt 2.1. Berechne $i = \lfloor \log[(\text{PREORDER}(v) - 1)$ **xor** $(\text{PREORDER}(v) + \text{SIZE}(v) - 1)] \rfloor$.

Schritt 2.2. Die Zusammensetzung von INLABEL(v) basiert auf folgenden zwei Beobachtungen:

(1) Die $l - i + 1$ linkesten Bits von INLABEL(v) stimmen mit den $l - i + 1$ linkesten Bits in PREORDER$(v)+$ SIZE$(v) - 1$ überein.

(2) Die restlichen Bits in INLABEL(v) sind alle Bits ”0”.

ASCENDANT Nummern

Idee: Zu jedem Knoten v erfasst ASCENDANT(v) die INLABEL Nummern aller Vorfahren von v in T.

Die INLABEL Nummer von jedem Vorfahren von v in T ist durch den Index des am weitesten rechts stehenden Bits ”1” eindeutig.

Sei $A_l(v), \cdots, A_0(v)$ die zugehörige Binärsequenz von ASCENDANT(v).

$A_i(v) = 1$, falls i der Index des am weitesten rechts stehenden Bits ”1” in der INLABEL Nummer eines Vorfahrens von v in T ist.

r sei die Wurzel von T. Setze ASCENDANT$(r) = 2^l$.

Betrachte einen inneren Knoten v in T. $F(v)$ sei Vater von v in T.

ASCENDANT$(v) =$ ASCENDANT$(F(v))$, falls

INLABEL$(v) =$INLABEL$(F(v))$.

ASCENDANT$(v) =$ ASCENDANT$(F(v)) + 2^i$, sonst.

i ist der Index des rechtesten Bits ”1” in INLABEL(v).

LEVEL(v) bezeichnet in einem Pfad die Anzahl der Kanten von einem Knoten v in T zur Wurzel r.

Bestimmung der Tabelle HEAD:

HEAD(k) beinhaltet denjenigen Knoten im Pfad mit der INLABEL Nummer k, der der Wurzel am nächsten ist.

HEAD(INLABEL(v)) wird auf v gesetzt, falls INLABEL$(v) \neq$ INLABEL$(F(v))$.

LCA Anfragen

Betrachte zu zwei Knoten x, y in T eine Anfrage LCA(x, y).

Fall A. INLABEL$(x) =$ INLABEL(y).

LCA(x, y) ist x, falls LEVEL$(x) \leq$ LEVEL(y) und y sonst.

Fall B. INLABEL$(x) \neq$ INLABEL(y). Sei z LCA(x, y). Wir finden z in vier Schritten:

Schritt 1. Finde b, LCA VON INLABEL(x) und INLABEL(y) im vollständigen Binärbaum B.

Schritt 2. Finde INLABEL(z).

Beobachtung: INLABEL(z) ist von allen Vorfahren von b, auf die Vorfahren von x und y in T abgebildet werden, der kleinste.

Schritt 3. Finde die Knoten $\hat{x}$ und $\hat{y}$. $\hat{x}$ ist der kleinste Vorfahre von x im INLABEL(z) zugehörigen Pfad und $\hat{y}$ ist der kleinste Vorfahre von y im gleichen Pfad. z ist von beiden Knoten derjenige, der der Wurzel näher ist.

Schritt 4. LCA(x, y) ist $\hat{x}$, falls LEVEL$(\hat{x})$ $\leq$LEVEL$(\hat{y})$, und $\hat{y}$ sonst.

Parallelisierung der Vorarbeit

Anwendung der Euler-Tour Technik auf den Baum T zur Bestimmung der Attribute der Knoten.

Anmerkung: Ein Pfad (Kreis) in einem Graphen (Digraphen) heisst *Eulerscher Pfad*, wenn er jede Kante genau einmal enthält.

Annahme: Alle Kanten in T sind zur Wurzel r gerichtet.

Vorgehen:

Schritt 1. Füge zu jeder Kante $(v \to u)$ in T eine Kante $(u \to v)$ in entgegengesetzter Richtung ein. H sei der neue Graph.

Schritt 2. Sei $D(e)$ die Nachfolgerkante von e im Eulerpfad. Mache zu jedem Knoten v in H folgendes: Seien $(v \to u_0), \cdots, (v \to u_{d-1})$ die von v ausgehenden Kanten.

$D(u_i \to v) := (v \to u_{(i+1) mod\ d}), i = 0, ..., d-1$. D hat nun einen Euler Kreis. ”Korrektur” $D(u_{d-1} \to r)$:=*end-of-list* liefert einen Eulerpfad mit Start- und Endknoten r.

Schritt 3. Jede Kante e im Eulerpfad bekommt zwei Gewichte $W_1(e)$ und $W_2(e)$ zugewiesen:

(1) $W_1(e) = 1$, falls e abwärts zeigt und $W_1(e) = 0$ sonst.

(2) $W_2(e) = 1$, falls e abwärts zeigt und $W_2(e) = -1$ sonst.

Schritt 4. Zweimalige Anwendung eines list-ranking Algorithmus, der zu jeder Kante e in H den gewichteten Abstand vom Start des Eulerpfades zu e liefert. Die Ergebnisse werden in DISTANCE$_1(e)$ und in DISTANCE$_2(e)$ gespeichert.

PREORDER(v) ist DISTANCE$_1(u \to v) + 1$,

PREORDER(v)+SIZE$(v) - 1$ ist DISTANCE$_1(v \to u) + 1$,

und LEVEL(v) ist DISTANCE$_2(u \to v)$.

Schritt 5. Falls zu jedem Knoten v in T PREORDER(v) und PREORDER(v)+SIZE$(v) - 1$ gegeben sind: Bestimmung von INLABEL(v) analog zu sequentiellem Algorithmus in konstanter Zeit. (Verwendung von n Prozessoren).

Schritt 6. Jede Kante e im Eulerpfad bekommt ein neues Gewicht $W(e)$. Sei u der Vater von einem Knoten v in T $(v \neq r)$ und sei i der Index der rechtesten "1" in INLABEL(v).

Falls INLABEL$(v) \neq$ INLABEL(u), $W(u \rightarrow v) \ = \ 2^i$ und $W(v \rightarrow u) \ = \ -2^i$. Das Gewicht der anderen Kanten wird auf Null gesetzt.

Schritt 7. Erneute Anwendung des list-ranking Algorithmus, der zu jeder Kante e in H deren gewichteten Abstand vom Start des Eulerpfades bestimmt. Betrachte einen Knoten $v \neq r$ und sei u sein Vater in T. ASCENDANT(v) ist der Abstand der Kante $(u \rightarrow v)$ plus 2^l. (ASCENDANT$(r) = 2^l$)

Falls die Attribute gegeben sind: Bestimmung der Tabelle HEAD in konstanter Zeit unter Verwendung von n Prozessoren.

Komplexität. Schritte 4 und 7 mit jeweils $n/\log n$ Prozessoren und $O(\log n)$ Laufzeit. Schritte 1,2,3,5,6 und HEAD mit jeweils n Prozessoren und $O(1)$ Zeit. Durchführung auch mit $n/\log n$ Prozessoren in $O(\log n)$ Zeit möglich.
Fazit: Parallelisierung der Vorarbeit in $O(\log n)$ Zeit mit $n/\log n$ Prozessoren. Beantwortung jeder LCA-Anfrage mit Hilfe eines einzelnen Prozessors in $O(1)$ Zeit.