Approximate Pattern Matching

Überblick:

- Motivation
- Edit-Distanz zweier Wörter
  - informelle Definition
  - Eigenschaften
  - Definition 2
  - dynamisches Programm
  - Edit-Graph
- Longest Common Subsequence
  - Definition
  - Zusammenhang mit der Edit-Distanz

Motivation

Oft nicht nur exakte, sondern auch „gute“ Übereinstimmung eines Musters in einem Text interessant:

- Textdatenbanken, z.B. Suchwort: *Shakespeare*

  Shakespears name is spelled in more different ways than that of any other english author.

  …

  Hat ***Shakespeer*** wirklich ***Shakespeers*** Dramen geschrieben?

  …

  Was ***Sheakspeare*** really in love?

- Computational Biology
  - Suche nach einem Genom in einer DNA Sequenz
  - Daten oft ungenau

=> Ähnlichkeitsbegriff für Strings benötigt!

informelle Def. der Edit-Distanz

- geg: zwei Wörter X, Y
- $edit(X,Y)$:= min. Anzahl Zeichen, die in X …
  - einzufügen (ins)
  - zu ersetzen (chng)
  - oder zu löschen (del)

  … sind, um Y zu erhalten

- Bsp.: *edit*(`Sh akespeare`,
  ins del del
  `Sheak spear`) = 3

  edit(`bla`, `blu`) = 1
  chng

Eigenschaften der Edit-Distanz

- evtl. existieren verschiedene minimale ins, chng, del Folgen
- sinvolle Def. eines Abstands, da:
  - $edit(X, Y) = 0 \iff X = Y$
  - $edit(X, Y) = edit(Y, X)$ Symmetrie
  - $edit(X, Y) \leq edit(X, Z) + edit(Z, Y)$ Dreiecksungl.

Definition 2

- geg: $X = X[1..m]$, $Y = Y[1..n]$

Bsp:	X	Y
	`bla`	`bub`

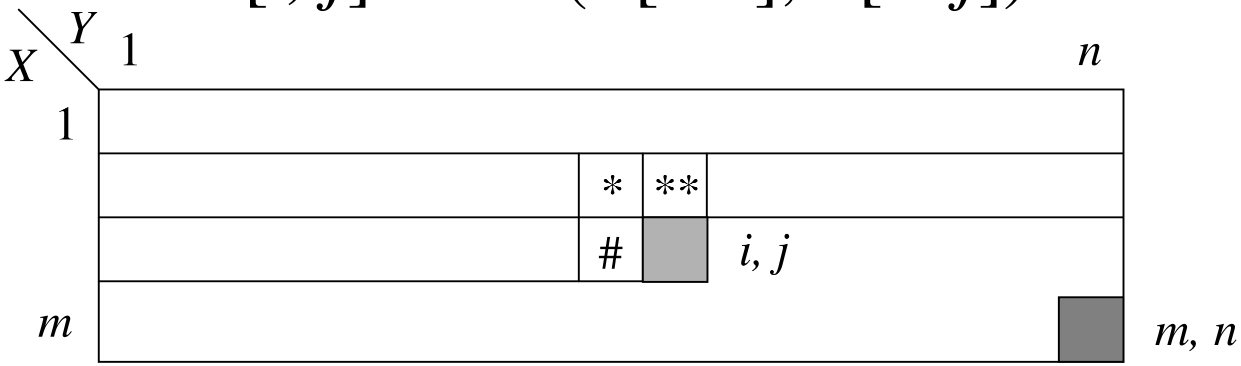
- Y schrittweise aus X aufbauen

Start: `bla`

- Operationen
  - cpy: akt. Zeichen an Y anfügen; bei X eins nach rechts rücken — `bla` `b`
  - del: akt. Zeichen überspringen — `bla` `b`
  - chng: neues Zeichen; eins weiter — `bla` `bu`
  - ins: neues Zeichen; nicht weiterrücken — `bla` `bub`
- ges: Folge mit min. Anzahl del, chng und ins Operationen; $edit(X,Y)$:= Anz. (äquiv. zu Def. 1)

Dynamisches Programm

- Tabelle $ED[i, j] = edit(X[1..i], Y[1..j])$ aufbauen



- betrachte $ED[i, j]$; letzte Op. einer opt. Folge war?

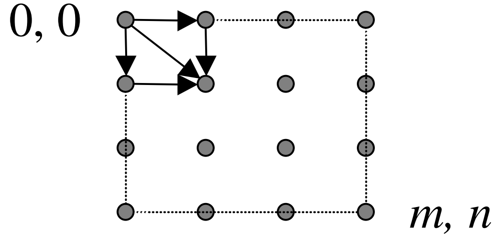
	1 i	1 j
– cpy: $ED[i, j] = ED[i\text{-}1, j\text{-}1]$ (*)	abaab	baba
– del: $ED[i, j] = ED[i\text{-}1, j] + 1$ (**)	abaab	baba
– chng: $ED[i, j] = ED[i\text{-}1, j\text{-}1] + 1$ (*)	abaab	baba
– ins: $ED[i, j] = ED[i, j\text{-}1] + 1$ (#)	abaab	baba

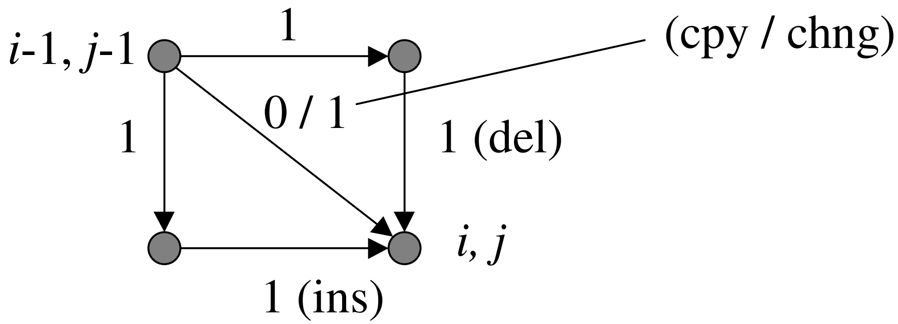
$ED[i, j] = \min(ED[i\text{-}1, j] + 1, ED[i, j\text{-}1] + 1, ED[i\text{-}1, j\text{-}1] + 0/1)$

- Zeit: $O(m\ n)$, bei geschickter Impl. Platz $O(\min(m, n))$

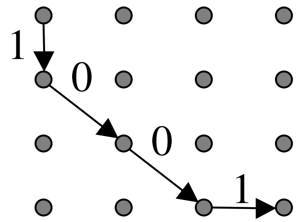
Edit-Graph

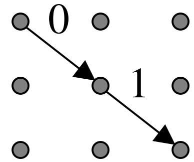
- anstelle des dyn. Programms ein Gitter aufbauen:





- Pfad entspricht einer Folge von Edit-Operationen
- ges: kürtzester Pfad von 0, 0 nach *m, n*
- DAG mit *m n* Knoten, *O*(*m n*) Kanten => LZ *O*(*m n*)
- Bsp.: `abb` `bbc` `bc` `bu`





Longest Common Subsequence - Def.

- geg: Wörter *X*, *Y*
- *longest*(*X*,*Y*) := Länge der längsten gem. Teilsequ. (nicht Teilstring: Lücken sind erlaubt!)
- Bsp: `Shakespeare` `Skakspeer`
- ges: Folge von cpy, del, ins Op. (kein chng!) mit min. Anzahl del, ins
- *longest*(*X*,*Y*) = Anzahl der cpy Operationen

Zusammenhang mit Edit-Distanz

- zur Erinnerung: $|X| = m$, $|Y| = n$
- $edit'(X, Y)$:= Edit-Distanz ohne chng Operationen
- c := Anzahl cpy Op. = $longest(X, Y)$
- $m - c$ = Anzahl del Op.
- $n - c$ = Anzahl ins Op.

=> $edit'(X, Y) = (m - c) + (n - c)$

=> $c = (m + n - edit'(X, Y)) / 2$