Informatik II: Algorithmen und Datenstrukturen SS 2013

Vorlesung 11b, Mittwoch, 3. Juli 2013
(Editierdistanz, dynamische Programmierung)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Editierdistanz
  - **Rekursiver** Algorithmus + Analyse
  - **Iterativer** Algorithmus + Analyse
  - Wie bekommt man die zugehörige Folgen von Operationen
  - Allgemeines Prinzip: **dynamische Programmierung**
  - Optimierungen des iterativen Algorithmus
  - Ü11, Aufgabe 2: Wie viele verschiedenen optimale Folgen von Operationen kann es geben? Obere Schranke + Beispiel mit möglichst vielen verschiedenen Folgen

Wiederholung

- Rekursive Formel zur Berechnung der ED
  - Für $|x| > 0$ und $|y| > 0$ ist:

    $ED(x, y)$ = das Minimum von

    - $ED(x[1..n], y[1..m-1]) + 1$, und (insert) — Fall 1A
    - $ED(x[1..n-1], y[1..m]) + 1$, und (delete) — Fall 1B
    - $ED(x[1..n-1], y[1..m-1] + 1$ falls $x[n] \neq y[m]$ (replace) — Fall 1C
    - $ED(x[1..n-1], y[1..m-1]$ falls $x[n] = y[m]$ — Fall 2
  - Für $|x| = 0$ ist $ED(\varepsilon, y) = |y|$
  - Für $|y| = 0$ ist $ED(x, \varepsilon) = |x|$

Rekursives Programm

- Es liegt nahe, das **rekursiv** zu programmieren
  - Für die Laufzeit gilt dann folgende rekursive Formel

    $T(n, m) \geq T(n-1, m) + T(n, m-1) + T(n-1, m-1) + \Omega(1)$
  - Dann $T(n, n) \geq 3^{n-1}$... also **exponentielle** Laufzeit

$T(n,m) \geq \underbrace{T(n-1,m)}_{\geq T(n-1,m-1)} + \underbrace{T(n,m-1)}_{\geq T(n-1,m-1)} + T(n-1,m-1) + A$

$\geq 3 \cdot T(n-1,m-1) + A$

$\geq 3 \cdot (3 \cdot T(n-2,m-2) + A) + A$

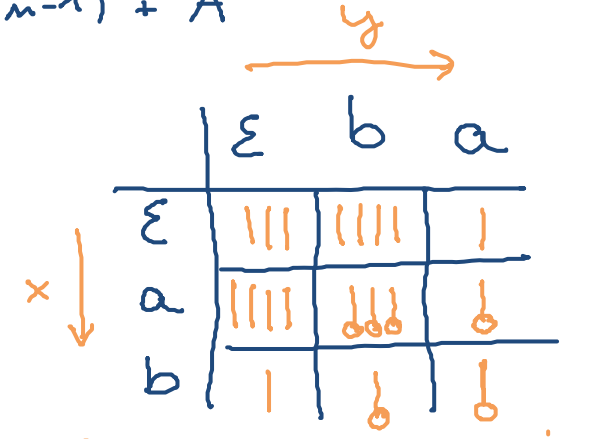
$\geq 3^2 \cdot T(n-2,n-2)$

$\dots$

$\geq 3^{n-1} \cdot T(1,1)$

1 = da zählt die Rekursion auf

1 = Zeit die FKT nochmal aufgerufen (2 x)

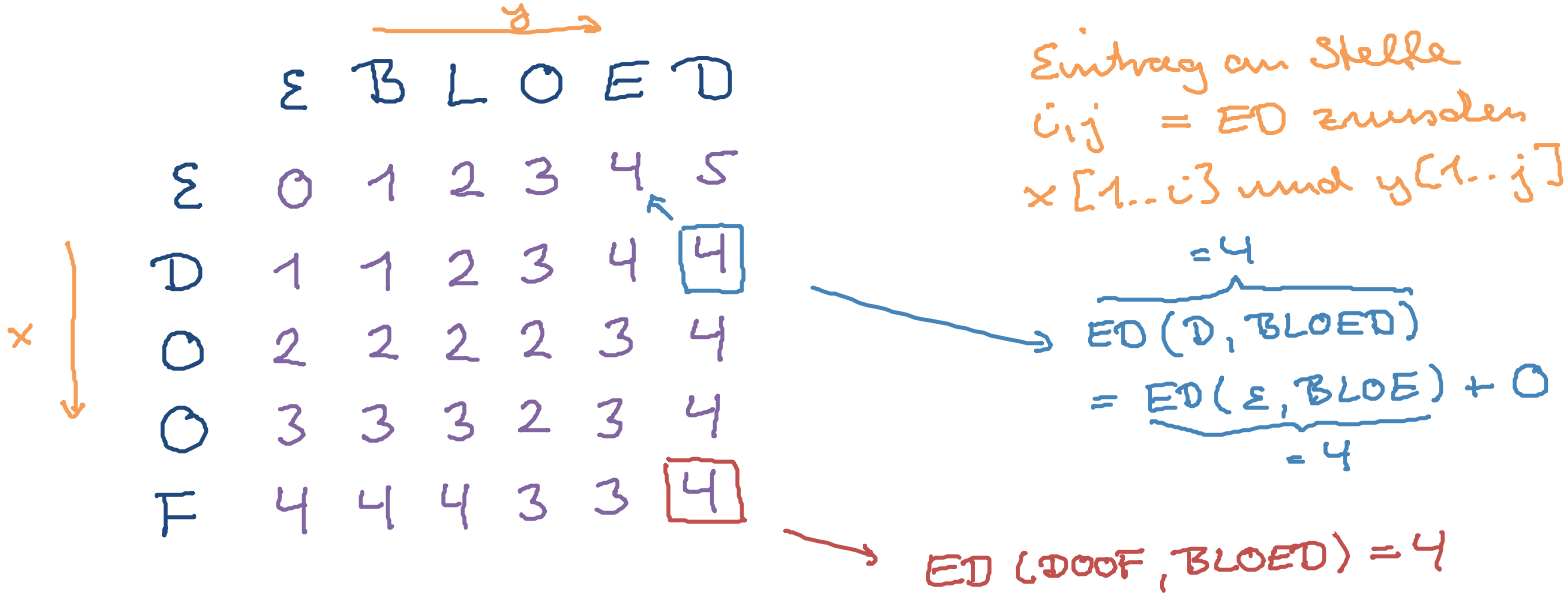


Eintrag an Stelle i,j gehört zur ED zwischen x[1..i] und y[1..j]

Iteratives Programm 1/9

$n = |x|$, $m = |y|$

- Wir können es auch **iterativ** machen
  - Wir merken uns alle Einträge, die wir schon berechnet haben … das sind ja nur $O(n \cdot m)$ viele



Iteratives Programm 2/9

UNI FREIBURG

- Laufzeitanalyse
  - Jeder Eintrag kann in Zeit $O(1)$ berechnet werden

    Für Zeile oder Spalte 0 trivial

    Sonst aus den drei benachbarten Einträgen
  - Es gibt genau $|x| \cdot |y|$ Einträge
  - Also insgesamt $O(|x| \cdot |y|)$ Zeit
  - Und ebenso $O(|x| \cdot |y|)$ Platz
  - Es geht auch schneller und mit weniger Platz ... nächste Folien

Iteratives Programm 3/9

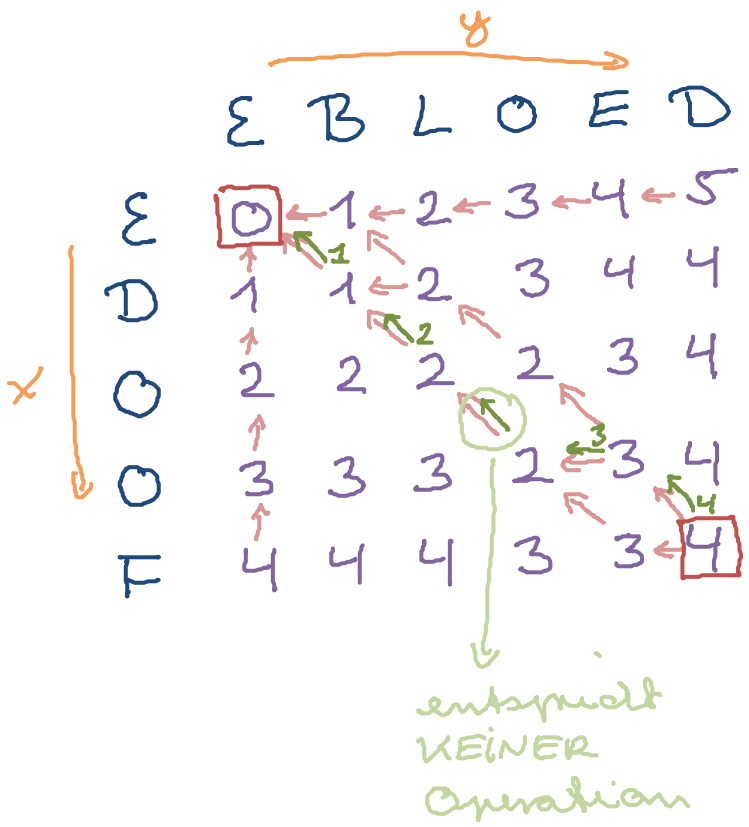
UNI FREIBURG

- Wie kommt man zu den Folgen von Operationen
  - Man merkt sich bei der Berechnung des Minimums der drei benachbarten Einträge, über welche das Minimum erzielt wurde

    Das können mehrere sein ! (eins, zwei oder drei)
  - Jeder Pfad von "unten rechts" (Eintrag bei |x|, |y|) nach "oben links" (Eintrag bei 0, 0) gibt einem dann eine mögliche Folge von Operationen
  - Man erhält so **alle** optimalen **monotonen** Folgen

    Das schauen wir uns jetzt anhand unseres Beispiels an

Iteratives Programm 4/9



← = woher das Minimum kommt

(nicht alle Pfeile gemalt sondern nur drei, die wir für Pfade von |x|, |y| nach 0,0 brauchen)

← ← ←
ein möglicher Weg
(es gibt insgesamt mehrere verschiedene)

D O O F ↓ REPLACE(1,B) ↖1
B O O F ↓ REPLACE(2,L) ↖2
B L O F ↓ INSERT(4,E) ←3
B L O E F ↓ REPLACE(5,D) ↖4
B L O E D

Iteratives Programm 5/9

UNI FREIBURG

- Wie viele verschiedene Folgen kann es geben ?
  - Das ist Ü11, Aufgabe 2
  - Hinweis: Sei k = ED(x, y), dann gibt es für jede der k Operationen drei Möglichkeiten, und die Positionen an denen diese stattfinden müssen eine aufsteigende Folge bilden

    Man muss sich überlegen, wie viele Möglichkeiten das insgesamt (höchstens) ergibt

Es geht um num-paths, nicht um $ED(x,y)$

da ist einfach $ED(x,y) \leq \min(|x|,|y|) + ||x|-|y||$ z.B.

$= \max(|x|,|y|)$

Iteratives Programm 6/9

UNI FREIBURG

- Optimierung Platzverbrauch
  - Beobachtung: am Ende interessiert einen nur der Eintrag "unten rechts" (an Stelle |x|, |y|)
  - Um den zu berechnen kann man auch Zeile für Zeile vorgehen, und sich nur die jeweils letzte Zeile merken

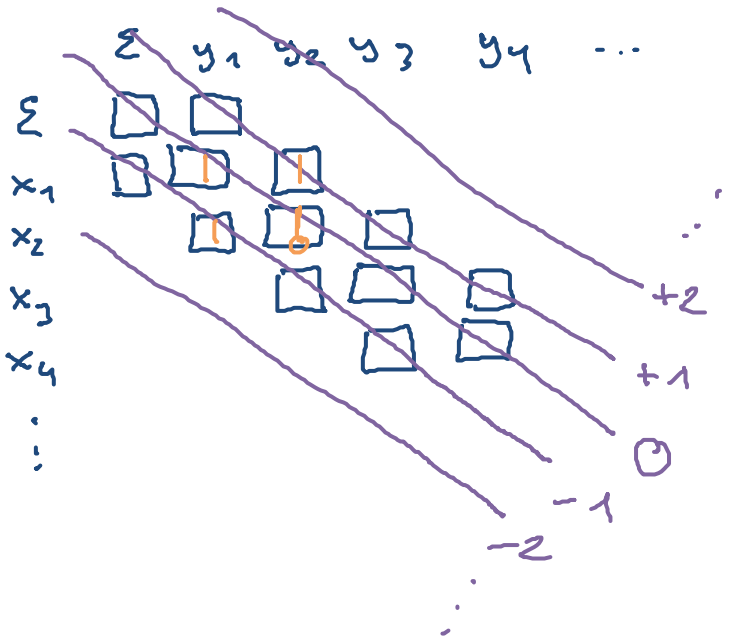
    Oder analog Spalte für Spalte berechnen, und sich dabei immer nur die jeweils letzte Spalte merken
  - Das benötigt dann nur Platz O(min(|x|, |y|))

    Falls |x| > |y| zeilenweise, sonst spaltenweise

Iteratives Programm 7/9

- Optimierung Laufzeit 1/3
  - Beobachtung: wenn man vorher weiß, dass die Editierdistanz höchstens δ groß ist …

    … dann reicht es, die Einträge auf der Hauptdiagonalen und den δ Nebendiagonalen darüber und darunter zu berechnen



Eintrag auf jeder Diagonalen lässt sich aus Einträgen auf diesr Diagonalen + den beiden Nachbar-diagonalen berechnen

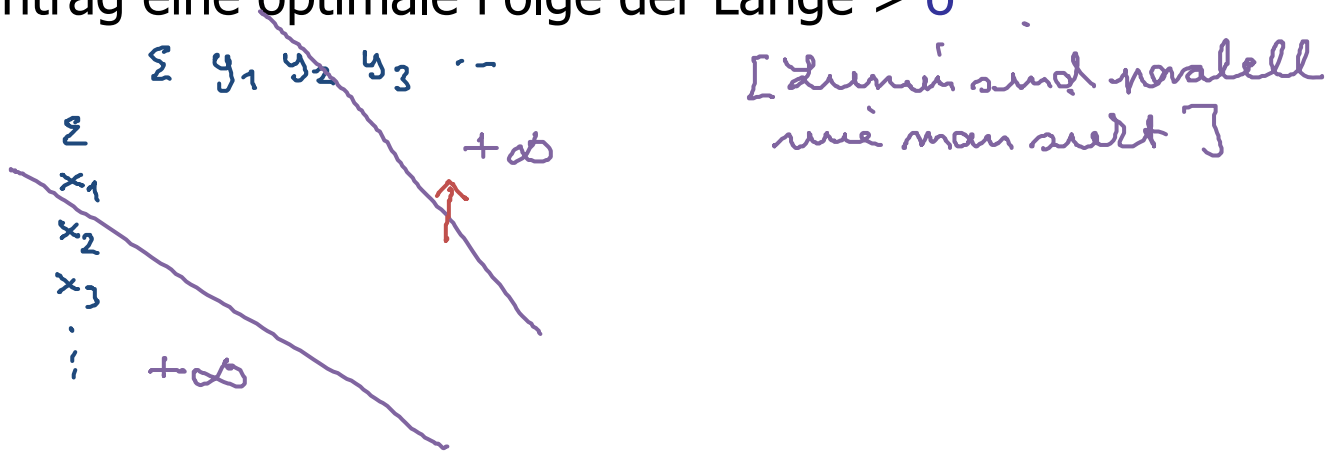
Iteratives Programm 8/9

UNI FREIBURG

- Optimierung Laufzeit 2/3
  - Wie berechnet man die Einträge am Rand des "Streifens" ?

    Die haben ja einen Nachbarn außerhalb des "Streifens"
  - Man kann die Einträge außerhalb des "Streifens" einfach auf $+\infty$ setzen

    Würde man nämlich (im Ursprungstableau) über einen dieser Einträge ein Minimum bekommen, bekäme man über diesen Eintrag eine optimale Folge der Länge > δ



Iteratives Programm 9/9

1+ ... damit da keine Null steht

UNI FREIBURG

- Optimierung Laufzeit 3/3
  - Jede der $2\delta + 1$ Diagonalen hat höchstens k Einträge, also Laufzeit $O(k \cdot (1 + \delta))$, wobei $k = \min(|x|, |y|)$
  - Aber woher bekommen wir ein gutes $\delta \geq ED(x, y)$?
  - Idee: wir probieren es einfach der Reihe nach für $\delta = 1, 2, 4, 8, 16, \ldots$ aus

$\delta = 1$

$\delta = 1, 2, 3, 4, \ldots$ wäre zu teuer!
dann Gesamtzeit $\sim ED(x,y)^2$

  - Sobald sich der Wert, der "unten rechts" herauskommt, nicht mehr ändert, hören wir auf
  - Lemma 1: wenn keine Änderung für größeres δ, dann ist der Wert unten rechts schon $ED(|x|, |y|)$

$ED = 11$
$\delta = 1, 2, 4, 8, \underline{16}, \underline{32}$
$\geq 11 \quad \geq 11$

  - Lemma 2: das δ bei dem man aufhört ist $\leq 4\ ED(x, y)$
  - Also Laufzeit insgesamt $O(\min(|x|, |y|) \cdot \underline{ED(x, y)})$

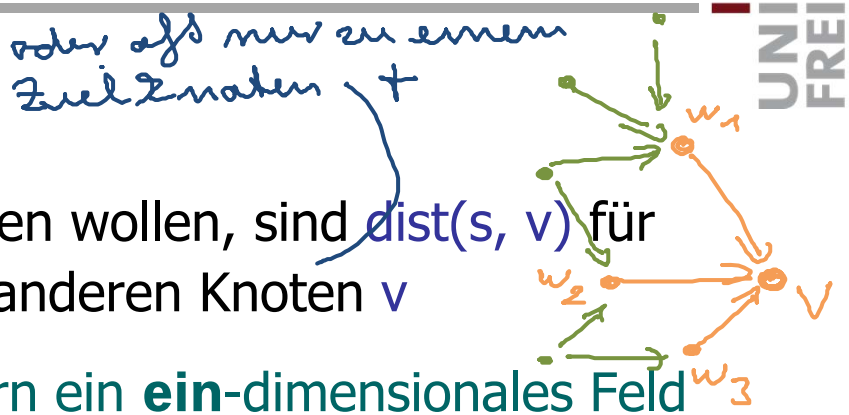
$1 + 2 + 4 + \ldots + 32 = 2 \cdot 32 - 1$

Dynamische Programmierung 1/2

UNI FREIBURG

- Allgemeines Prinzip
  - Man hat eine **rekursive** Formel, bei der dieselben Teilprobleme mehrfach vorkommen (in verschiedenen rekursiven Aufrufen)
  - Man berechnet dann **iterativ**, in einer systematischen Reihenfolge, die Lösung aller relevanten Teilprobleme
  - Zusammen mit dem "Wert" der optimalen Lösung erhält man so immer auch den "Weg" dorthin
  - Der Name "dynamische Programmierung" ist ziemlich irreführend und hat wenig mit dem Grundprinzip zu tun
  - Dijkstras Algorithmus war auch ein (verkapptes) Beispiel für dynamische Programmierung ... siehe nächste Folie

Dynamische Programmierung 2/2



- Dijkstras Algorithmus
  - Die Einträge, die wir berechnen wollen, sind dist(s, v) für einen Startknoten s und alle anderen Knoten v

    In dem Fall kein zwei- sondern ein **ein**-dimensionales Feld
  - Rekursive Formel: dist(s, v) = min dist(s, w) + cost(w, v) wobei das Minimum über alle nach v eingehenden Kanten (w, v) gebildet wird

    Würde man das trivial rekursiv programmieren, hätte man das gleiche Problem wie beim rekursiven Programm für ED
  - Stattdessen macht man es **iterativ**, in der Reihenfolge aufsteigender Entfernung (bezüglich dist Wert) von s

    Reihenfolge aber aufwändiger zu realisieren als bei ED !

Literatur / Links

UNI FREIBURG

- Dynamische Programmierung
  - In Mehlhorn / Sanders

    12.3 Dynamic Programming
  - In Wikipedia

    http://en.wikipedia.org/wiki/Dynamic_programming

    http://de.wikipedia.org/wiki/Dynamische_Programmierung