Algorithmen und Datenstrukturen (ESE)
Entwurf, Analyse und Umsetzung von Algorithmen (IEMS)
WS 2012 / 2013

Vorlesung 14, Dienstag, 5. Februar 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

- Organisatorisches
  - Ihre Erfahrungen + Ergebnisse mit dem Ü13 (Dijkstra)
  - Offizielle **Evaluation** dieser Vorlesung
- Editierdistanz
  - Maß für Ähnlichkeit zwischen zwei Wörtern / Zeichenketten
  - Algorithmus zur effizienten Berechnung
  - Allgemeines Prinzip: dynamische Programmierung
  - **Übungsblatt 14:** Implementierung des Algorithmus + damit ähnliche Suchanfragen im AOL Query Log finden

Erfahrungen mit dem Ü13 (Dijkstra)

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 5. Februar 15:00
  - Schöne Aufgabe, gut motiviert
  - Aber nicht einfach, es korrekt hinzubekommen
  - Zeitaufwändig, auch wegen Reparaturen an der Ü12 Lösung
  - Etwas tricky, den Graphen auf seine LCC zu reduzieren
  - Visualisierung auf Google Maps war gut
  - Dijkstra Weg als KML Datei ausgegeben !
  - Musterlösung in C++ bringt den Java Leuten nichts

Offizielle Evaluation der Vorlesung

UNI FREIBURG

- Bitte den Bogen **bis Ende der Woche** abgeben
  - Ich werde das Feedback dann nächste Woche (= letzte Vorlesung) zusammenfassen und besprechen !
  - Sie bekommen dafür 20 wunderschöne Punkte
  - Schreiben Sie dazu einfach in Ihre erfahrungen.txt, dass Sie den Evaluationsbogen ausgefüllt haben (wenn es so ist)
  - Nehmen Sie sich bitte genug Zeit für das Ausfüllen
  - Die Freitextkommentare sind für uns am interessantesten
  - Seien Sie bitte ehrlich und möglichst konkret
  - **Abgabe bitte bis Freitag diese Woche (8. Februar)**
    … und allerspätestens bis Sonntag (10. Februar)

Motivation

UNI FREIBURG

- Viele Anwendungen, wo man ähnliche Strings sucht
  - Dubletten in Adressdatenbanken

    Hein Blöd, 27568 Bremerhaven

    Hein Bloed, 27568 Bremerhafen

    Hein Doof, 27478 Cuxhaven
  - Produktsuche

    Memory Stik
  - Websuche

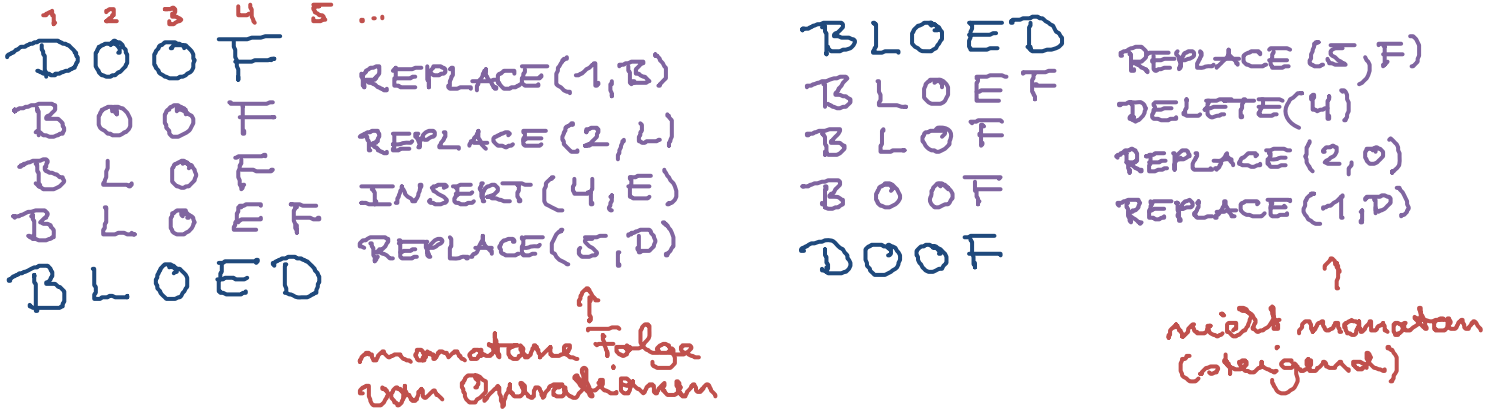
    eyjaföllajaküll

    uniwersität verien 2013
  - Gemeinsamkeit: man braucht ein Maß für die **Ehnlichkeit** zwischen zwei Strings

Editierdistanz 1/9

UNI FREIBURG

- Definition Editierdistanz, auch Levenshtein-Distanz
  - Gegeben zwei Zeichenketten (strings) x und y
  - ED(x, y) = Editierdistanz (edit distance) von x und y = die minimale Anzahl Operationen um x in y zu transformieren:
    - Einfügen eines Buchstabens (insert)
    - Ersetzen eines Buchstabens durch einen anderen (replace)
    - Löschen eines Buchstabens (delete)
    - Die Position einer Operation ist ... siehe Beispiel:



Editierdistanz 2/9

UNI FREIBURG

- Etwas Notation
  - Mit ε bezeichnen wir das leere Wort
  - Mit $|x|$ bezeichnen wir die Länge von x (= Anzahl Zeichen)
  - Mit $x[i..j]$ bezeichnen wir die Teilfolge der Zeichen i bis j der Zeichenkette x, wobei $1 \leq i \leq j \leq |x|$
- Ein paar einfache Eigenschaften
  - $ED(x, y) = ED(y, x)$
  - $ED(x, \varepsilon) = |x|$
  - $ED(x, y) \geq abs(|x| - |y|)$ $\quad abs(x) = x > 0 \; ? \; x : -x$
  - $ED(x, y) \leq ED(x[1..n-1], y[1..m-1]) + 1$ $\quad n = |x|, m = |y|$

Editierdistanz 3/9

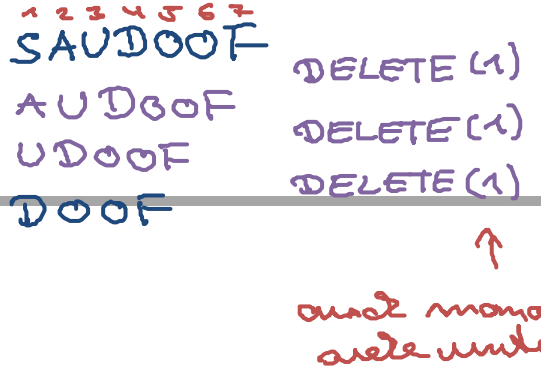
UNI FREIBURG

- Lösungsideen anhand von Beispielen
  - Für VERIEN → FERIEN ? ED = 1
  - Für MEXIKO → AMERIKA ? ED = 3
  - Für AAEBEAABEAREEAEBA → RBEAAEEBAAAEBBAEAE ?
  - Beobachtung: möglichst große gemeinsame Teilstrings zu finden klappt manchmal, aber nicht immer
- Rekursiver Ansatz
  - In zwei Teile / Hälften teilen? Keine gute Idee, z.B.

    ED(GRAU, RAUM) = 2 aber ED(GR, RA) + ED(AU, UM) = 4
  - Auf ein "kleineres" Problem zurückführen?

    Das probieren wir jetzt !

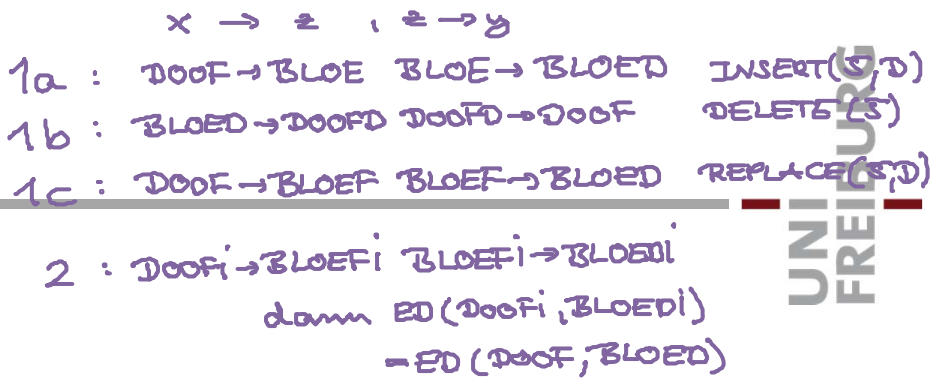
Editierdistanz 4/9



- Terminologie
  - Seien x und y unsere beiden Zeichenketten
  - Seien $\sigma_1, \ldots, \sigma_k$ eine Folge von $k = ED(x, y)$ Operationen für $x \rightarrow y$, das heißt um x in y zu überführen

    (Wir nehmen im Folgenden nicht an, dass wir die Folge schon kennen, sondern nur, dass es so eine gibt)
  - Wir betrachten im Folgenden nur **monotone** Op.-Folgen, d.h. die Position von σ_{i+1} ist $\geq$ die Position von σ_i, wobei = nur dann erlaubt ist, wenn beides delete Operationen sind
  - **Lemma:** Für beliebige x und y mit $k = ED(x, y)$ gibt es eine **monotone** Folge von k Operationen für $x \rightarrow y$
  - Beweisintuition: die Reihenfolge der Operationen ist im Grunde egal, also kann man sie auch monoton anordnen

Editierdistanz 5/9



- Fallunterscheidung
  - Wir betrachten die letzte Operation σ_k
    - $\sigma_1, \ldots, \sigma_{k-1} : x \rightarrow z$ und $\sigma_k : z \rightarrow y$ (DOOF BLOEF BLOEF BLOED)
    - Seien $n = |x|$ (4) und $m = |y|$ (5) und $m' = |z|$ (5)
    - Man beachte, dass $m' \in \{m - 1, m, m + 1\}$ wieso?
  - Fall 1: σ_k macht etwas "ganz am Ende" von z, d.h. eins von:
    - Fall 1a: $\sigma_k = \text{insert}(m' + 1, y[m])$ [dann ist $m' = m - 1$]
    - Fall 1b: $\sigma_k = \text{delete}(m')$ [dann ist $m' = m + 1$]
    - Fall 1c: $\sigma_k = \text{replace}(m', y[m])$ [dann ist $m' = m$]
  - Fall 2: σ_k macht nichts "ganz am Ende" von z
    - dann $z[m'] = y[m]$ und $x[n] = z[m']$ und damit

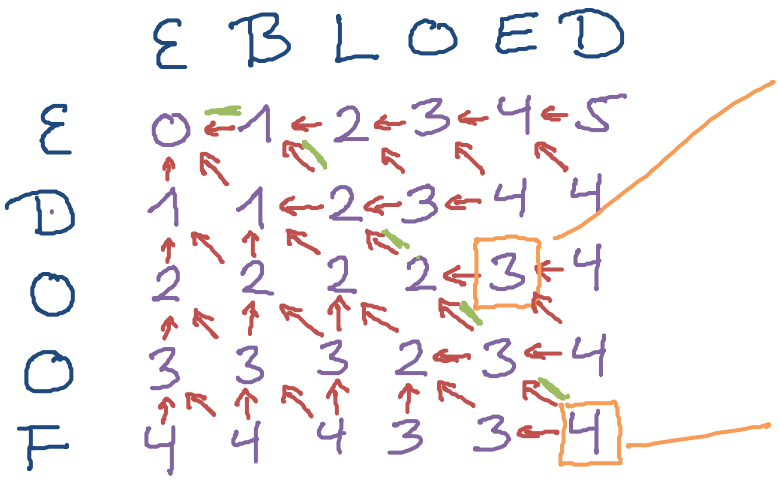
      $\sigma_1, \ldots, \sigma_k : x[1..n-1] \rightarrow y[1..m-1]$ und $x[n] = y[m]$

Editierdistanz 6/9

UNI FREIBURG

- Wir haben also einen dieser vier Fälle
  - Fall 1a (insert am Ende): $\sigma_1, \ldots, \sigma_{k-1} : x[1..n] \rightarrow y[1..m-1]$
  - Fall 1b (delete am Ende): $\sigma_1, \ldots, \sigma_{k-1} : x[1..n-1] \rightarrow y[1..m]$
  - Fall 1c (replace am Ende): $\sigma_1, \ldots, \sigma_{k-1} : x[1..n-1] \rightarrow y[1..m-1]$
  - Fall 2 (nichts am Ende): $\sigma_1, \ldots, \sigma_k : x[1..n-1] \rightarrow y[1..m-1]$
- Daraus folgt die rekursive Formel
  - Für $|x| > 0$ und $|y| > 0$ ist $ED(x, y)$ = das Minimum von
    - $ED(x[1..n], y[1..m-1]) + 1$, und
    - $ED(x[1..n-1], y[1..m]) + 1$, und
    - $ED(x[1..n-1], y[1..m-1] + 1$ falls $x[n] \neq y[m]$
    - $ED(x[1..n-1], y[1..m-1]$ falls $x[n] = y[m]$
  - Für $|x| = 0$ ist $ED(x, y) = |y|$, für $|y| = 0$ ist $ED(x, y) = |x|$

Editierdistanz 7/9



das ist ED(DO, BLOE)
der rote Pfeil sagt uns
ED(DO, BLOE) = ED(DO, BLO) + 1
und die Operation war
INSERT(4, E)

das ist ED(DOOF, BLOED)

Folge von Operationen

DOOF
BDOOF
BLOOF
BLOEF
BLOED

INSERT(1, B)
REPLACE(2, L)
REPLACE(4, E)
REPLACE(5, D)

wie viele versch. Folgen gibt es?

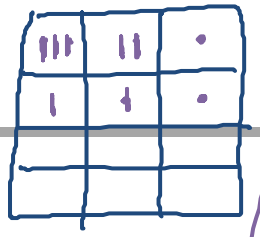
Bemerkung: nach diesem Schema findet man nur die minimalen Folgen (aber das reicht ja)

Editierdistanz 8/9

UNI FREIBURG

- Wie bekommen wir die Folge von Operationen?
  - Wir merken uns einfach bei jeder Anwendung der Rekursionsformel, welcher der vorherigen Einträge den kleinsten Wert ergeben hat (die Pfeile in unserem Bild)
    - Es kann von einem Eintrag mehrere Pfeile zu den drei Einträgen davor geben
  - Wenn wir den Pfeilen von dem Eintrag bei (n, m) bis zum Eintrag für (1, 1) folgen, bekommen wir eine optimale Folge von Operationen
    - Können wir unterwegs mehreren Pfeilen folgen, gibt es entsprechend mehrere optimale Folgen
    - Diese Folgen sind nach Konstruktion alle monoton

Editierdistanz 9/9



$T(m,m) \geq 3T(m-1,m-1) \geq 3^m$

- Rekursives Programm
  - Es liegt nahe, das rekursiv zu programmieren
  - Für die Laufzeit würde folgende rekursive Formel gelten

    $T(n, m) = T(n-1, m) + T(n, m-1) + T(n-1, m-1) + 1$

    $\geq T(m-1,m-1) \quad \geq T(m-1,m-1)$
  - Man kann leicht ausrechnen, dass dann $T(n, n) \geq 3^n$
  - Das heißt die Laufzeit wäre (mindestens) exponentiell
- Dynamische Programmierung
  - Wir berechnen die Tabelle einfach Eintrag für Eintrag, so wie wir es in dem Beispiel eh gemacht haben und merken uns alle Einträge, die wir schon berechnet haben
  - Das braucht dann Laufzeit und Speicherplatz $O(n \cdot m)$

Dynamische Programmierung

UNI FREIBURG

- Das allgemeine Prinzip dazu
  - Rekursive Berechnung, bei der

    … dieselben Teilprobleme mehrmals auftauchen

    … die Gesamtzahl von Teilproblemen begrenzt
  - Dann Lösung für alle Teilproblem berechnen
  - Und zwar nach und nach in solch einer Reihenfolge

    … dass sich noch nicht berechnete Lösungen aus schon berechneten zusammensetzen lassen
  - Zusammen mit dem "Wert" der optimalen Lösung erhält man so in der Regel auch den "Weg" dorthin
  - Dijkstras Algorithmus war vom Prinzip her auch dynamische Programmierung !

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
- Editierdistanz
  - In Wikipedia

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

    http://de.wikipedia.org/wiki/Levenshtein-Distanz