Informatik II: Algorithmen und Datenstrukturen SS 2015

Vorlesung 11a, Dienstag, 7. Juli 2015
(Editierdistanz, Teil 1)

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 mit dem Ü10 (BFS, CC, Dijkstra)
  - Offizielle Evaluation der Veranstaltung
  - Wann ist ein Berg ein Berg
- Inhalt
  - Editierdistanz … Motivation, Definition, rekursive Formel
  - Ü11, Aufgabe 1: Online-Evaluation (20 Punkte !!!)

    Zur Aufgabe 2 in der Vorlesung morgen mehr

Erfahrungen mit dem Ü10 1/4

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 7. Juli 12:00
  - Aufgabe 1 (Laufzeit) gute Übung aber fiel vielen schwer

    Siehe Erklärungen auf den nächsten drei Folien
  - Aufgabe 2 (Dijkstra) spannend aber für viele auch schwieriger als erwartet (in den Details)
  - Wie erwartet hatten einige Probleme mit BaWü
  - Mein längster kürzester Pfad ist zu kurz
  - Zu … heiß … zum … Aufgaben Machen
  - Abschließende Notizen auf dem Ü immer kryptischer, hoffentlich nicht klausur-elefant
  - Rosskopf: Prominenz ca. 100m, Dominanz ca. 4km

Erfahrungen mit dem Ü10 2/4

UNI FREIBURG

und noch: Rekursion

- Laufzeitbestimmung, Vorbetrachtung
  - Grundsätzliches Vorgehen: die **Schleifen** anschauen

    Nur da kann nicht-konstante Laufzeit verbraten werden

    Dabei spielt es keine Rolle, ob es eine for Schleife oder eine while Schleife ist, wichtig ist, wie oft sie durchlaufen wird
  - Achtung bei Funktionsaufrufen: innerhalb der Funktion können sich auch Schleifen befinden

    Wenn man die Funktion nicht kennt, in der Doku nach Laufzeit schauen … sonst plausible Annahmen machen

    Zum Beispiel: beim Zugriff auf ein Element von einem Feld kann man davon ausgehen, dass es in O(1) Zeit geht

Erfahrungen mit dem Ü10 3/4

UNI FREIBURG

- Lösungsskizze Aufgabe 1
  - In unserem Code in bfs.py gibt es zwei Schleifen ... alle anderen Operationen haben konstante Laufzeit

    Die äußere Schleife hat insgesamt konstante Kosten für jeden Knoten in jedem Level

    Die innere Schleife hat insgesamt konstante Kosten für jede Kante, die von einem dieser Knoten ausgeht
  - Durch das "visited" wird jeder Knoten und jede Kante nur **genau einmal** angeschaut
  - Die Laufzeit ist also $\Theta(n' + m')$, wobei n' und m' die Anzahl Knoten und Kanten in der Zusammenhangskompomente sind, in der sich der Startknoten befindet

bei ungerichteten Graphen: jede Kante 2 Knoten zweimal

Erfahrungen mit dem Ü10 4/4

UNI FREIBURG

- Laufzeitbestimmung, Nachbetrachtung
  - Es gibt zwar Regeln (siehe Folie 4), aber die Laufzeit-bestimmung ist kein einfacher schematischer Prozess

    Vergleichbar mit einem mathematischen Beweis

    In der Tat kann Laufzeitbestimmung beliebig komplex sein, selbst für einfache Algorithmen

    Für BFS war es aber relativ einfach, und so wird es auch sein, wenn es in der Klausur dran kommt

Offizielle Evaluation der Veranstaltung

UNI FREIBURG

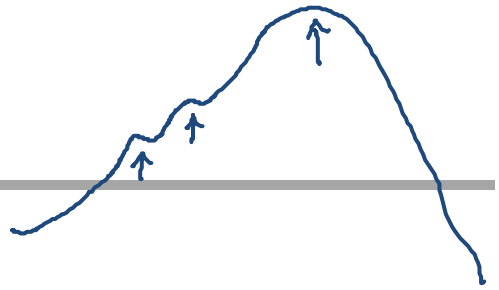
- Läuft über das zentrale **EvaSys** der Uni
  - Sie sollten gestern (Montag, 6. Juli) eine Mail vom System bekommen haben

    Falls nicht, bitte umgehend bei uns melden !
  - Nehmen Sie sich bitte Zeit und füllen Sie den Bogen sorgfältig und gewissenhaft aus

    Sie haben soviel Zeit in die Vorlesung investiert, dann können Sie auch 30 Minuten für die Evaluation aufwenden

    Sie bekommen außerdem 20 wunderschöne Punkte dafür
  - Uns interessieren besonders die Freitextkommentare

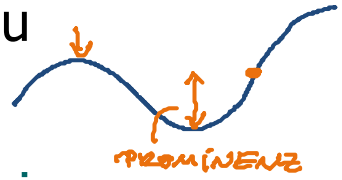
Wann ist ein Berg ein Berg



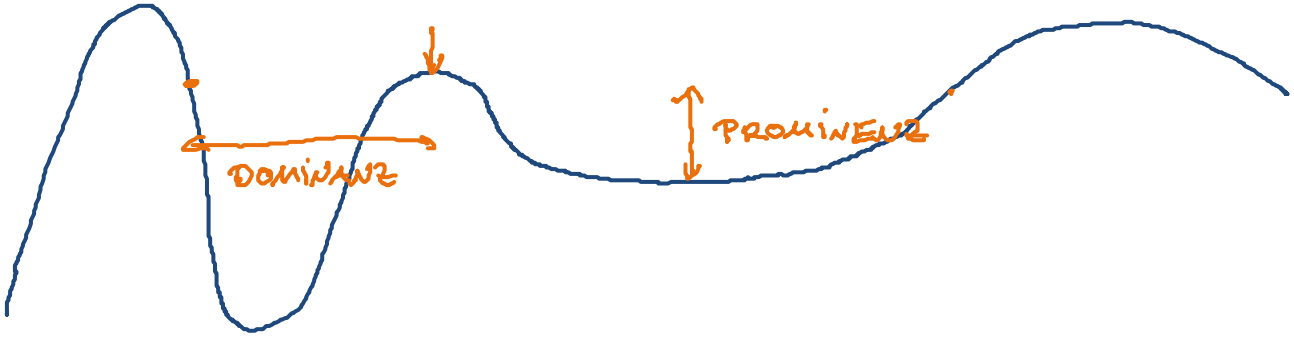
- Dominanz und Prominenz
  - Dominanz = Entfernung (Luftlinie) zum nächstgelegen Punkt gleicher Höhe

    Intuitiv: wie weit der nächste höhere Berg weg ist
  - Prominenz = minimaler Abstieg (Höhenmeter), um zu einem anderen Punkt gleicher Höhe zu kommen

    Intuitiv: niedrigster Graben zu einem höheren Berg hin
  - 1-Millionen-Euro Frage: kann dieser höhere Berg ein anderer sein für Dominanz und Prominenz ?



Editierdistanz 1/16

UNI FREIBURG

- Motivation
  - Es gibt viele Anwendungen, wo man ein Maß für die Ähnlichkeit zwischen zwei Zeichenketten braucht

    Drei Beispiele dazu auf der nächsten Folie
  - Die **Editierdistanz** ist ein solches Ähnlichkeitsmaß

    Das in der Praxis am häufigsten verwendete Maß

    Definition auf der übernächsten Folie

    Benannt nach dem Zeichenketten fressenden sogenannten **Edi-Tier**

Editierdistanz 2/16

UNI FREIBURG

- Anwendungsbeispiele
  - Beispiel 1: Dubletten in Adressdatenbanken

    Hein Blöd, 27568 Bremerhaven
    Hein Bloed, 27568 Bremerhafen
    Hein Doof, 27478 Cuxhaven

  - Beispiel 2: Produktsuche

    Memori Stik

  - Beispiel 3: Websuche

    eyjaföllajaküll
    uni semesta verien 2015

Editierdistanz 3/16

UNI FREIBURG

- Definition Editierdistanz … alternativ: Levenshtein-Distanz
  - Gegeben zwei Zeichenketten x und y
  - ED(x, y) = die minimale Anzahl der folgenden Operationen, die man braucht, um x in y zu transformieren:

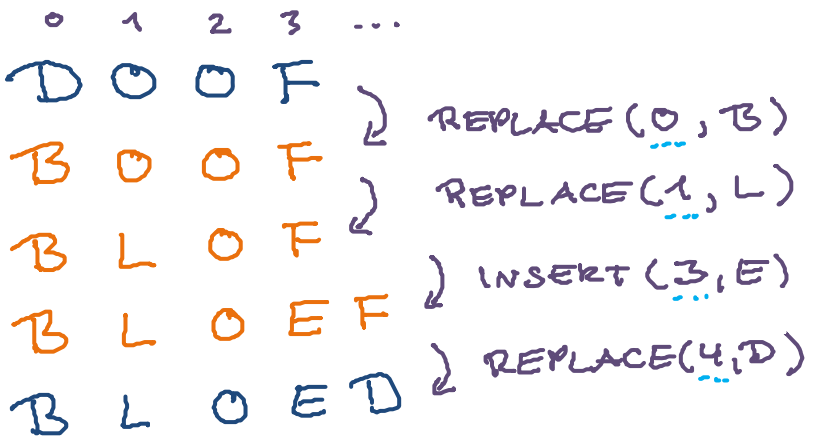
    Einfügen eines Buchstabens (insert)

    Ersetzen eines Buchstabens durch einen anderen (replace)

    Löschen eines Buchstabens (delete)
  - Die Position einer Operation ist die Stelle im String, an der etwas geändert wird … siehe Beispiel nächste Folie

Editierdistanz 4/16

- Beispiel DOOF → BLOED ED = ?



Diese Folge ist MONOTON (s. Folie 18)

0 < 1 < 3 < 4

Das zeigt jetzt erstmal nur, dass ED ≤ 4

(Es könnte ja auch noch mit weniger gehen)

Editierdistanz 5/16

UNI FREIBURG

- 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 $0 \leq i \leq j < |x|$

$x =$ BLOED (0 1 2 3 4) $\quad x[1..3] =$ LOE $\quad |x| = 5$

Editierdistanz 6/16

UNI FREIBURG

- Ein paar einfache Eigenschaften
  - ED(x, y) = ED(y, x)
  - ED(x, ε) = |x|
  - ED(x, y) ≥ abs(|x| - |y|) abs(x) = x > 0 ? x : -x
  - ED(x, y) ≤ ED(x[1..n-1], y[1..m-1]) + 1 n = |x|, m = |y|

ED(x,y) = ED(y,x)

Es gibt zu jeder Folge $x \to \dots \to y$ eine entsprechende Folge $y \to \dots \to x$ die dasselbe rückwärts macht

REPLACE ↔ REPLACE

INSERT ↔ DELETE

DELETE ↔ INSERT

Editierdistanz 7/16

UNI FREIBURG

- Lösungsidee 1: möglichst viel "erhalten"
  - ED("VERIEN", "FERIEN") = 1

    Einfach, weil die Zeichenketten größtenteils gleich
  - ED("SEMESTERFERIEN", "SEMESTERVERIEN") = 1

    Dito … dann auch für längere Zeichenketten noch einfach
  - ED("MEXIKO", "AMERIKA") = 3

    Auch hier gibt es noch relativ große Übereinstimmung
  - ED("AAEBEAABEAREEAEBA", "RBEAAEEBAAAEBBAEAE") = 2.

    Spätestens hier wird es sehr schwierig mit dieser Idee

Editierdistanz 8/16

UNI FREIBURG

- Lösungsidee 2: in zwei Hälften teilen
  - In zwei gleich große Hälften teilen und die dann jeweils rekursiv lösen

ED(GRAU, RAUM) = 2

ED(GR, RA) = 2

ED(AU, UM) = 2

$x = x_1 x_2 \quad y = y_1 y_2$

Dann im Allgemeinen

$ED(x,y) \neq ED(x_1,y_1) + ED(x_2,y_2)$

Keine gute Idee, die ED zwischen den Hälften hat nicht viel zu tun mit der ED zwischen den ursprünglichen Zeichenketten

Editierdistanz 9/16

UNI FREIBURG

- Lösungsidee 3: alternative rekursive Formel
  - Das Problem auf ein Problem "eins kleiner" zurückführen

    ED("FERIEN", "VERIEN") = ED("FERIE", "VERIE")

    Weil die Worte rechts mit dem selben Buchstaben aufhören

    ED("SAUM", "RAUS") = ED("SAU", "RAU") + 1

    Weil eine optimale Transformation ein replace am Ende macht

    ED("RAUM", "GRAU") ≠ ED("RAU", "GRA") + 1

    Das gilt aber nicht immer, wenn sich die letzten beiden Buchstaben unterscheiden

    Mit einer etwas komplizierteren Formel kriegt man es aber hin, das sehen wir jetzt auf den nächsten Folien

Editierdistanz 10/16

SAUDOOF → → BUDOOF

- Monotonie
  - 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
  - Eine Folge von Operationen heißt **monoton**, wenn die Position von σ_{i+1} ist ≥ die Position von σ_i, wobei = nur dann erlaubt ist, wenn beides delete Operationen sind

    Eine Folge von delete Operationen mit gleichen Positionen braucht man zum Beispiel bei ED("saudoof", "doof")

Editierdistanz 11/16

UNI FREIBURG

- Hilfssatz zur Monotonie
  - **Lemma:** Für beliebige x und y mit $k = ED(x, y)$ gibt es eine monotone Folge von k Operationen für $x \rightarrow y$
  - Den Beweis lassen wir hier weg

    Intuition: die Reihenfolge der Operationen ist im Grunde egal, also kann man sie auch monoton anordnen

Editierdistanz 12/16

Beispiele für die drei U-Fälle
Fall 1a: DOOF → BLOE → BLOED (σ_1, σ_3 / σ_4)
Fall 1b: TROTTEL → IDIOTT → IDIOT (σ_1–σ_5 / σ_6)
Fall 1c: DEPP → NERP → NERD (σ_1, σ_2 / σ_3)

UNI FREIBURG

- Fallunterscheidung
  - Wir betrachten die letzte Operation σ_k

z.B. $k = 4$ DOOF BLOEF BLOEF BLOED

$\sigma_1, \ldots, \sigma_{k-1} : x \to z$ und $\sigma_k : z \to y$

z.B. Seien $n = |x|$ (4) und $m = |y|$ (5) und $m' = |z|$ (5)

Man beachte, dass $m' \in \{m - 1, m, m + 1\}$

  - **Fall 1:** σ_k macht etwas "ganz am Ende" von z :

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 $m' = m$]

Eine andere Möglichkeit als die drei gibt es nicht

Indizes fangen hier mit 1 an und in der Folge

Editierdistanz 13/16

UNI FREIBURG

Beispiel zu Fall 2:
DOOFI → BLOED!
$\sigma_1 - \sigma_4$

- Fallunterscheidung
  - Wir betrachten die letzte Operation σ_k

    $\sigma_1, \ldots, \sigma_{k-1} : x \to z$ und $\sigma_k : z \to y$

    Seien $n = |x|$ und $m = |y|$ und $m' = |z|$

    Man beachte, dass $m' \in \{m - 1, m, m + 1\}$
  - **Fall 2:** σ_k macht nichts "ganz am Ende" von z

    Dann $z[m'] = y[m]$ und $x[n] = z[m']$

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

Editierdistanz 14/16

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

Editierdistanz 15/16

- Daraus folgt die folgende rekursive Formel
  - Für $|x| > 0$ und $|y| > 0$ ist:

    $ED(x, y)$ = das Minimum von
    - $ED(x[1..n], y[1..m-1]) + 1$ für das „insert" am Ende
    - $ED(x[1..n-1], y[1..m]) + 1$ für das „delete" am Ende
    - $ED(x[1..n-1], y[1..m-1] + \delta$ für das „replace" am Ende

      wobei $\delta = 1$ falls $x[n] \neq y[m]$, sonst $\delta = 0$
  - Für $|x| = 0$ ist $ED(x, y) = |y|$
  - Für $|y| = 0$ ist $ED(x, y) = |x|$

Editierdistanz 16/16

UNI FREIBURG

- Rekursive Implementierung
  - Mit der Formel von der Folie vorher können wir jetzt sehr leicht ein rekursives Programm schreiben

    Es funktioniert auch (immerhin)

    Aber es dauert unverhältnismäßig lange, selbst schon für relative kurze Zeichenketten

Literatur / Links

- Editierdistanz
  - In Wikipedia (Definition + Algorithmen)

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

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