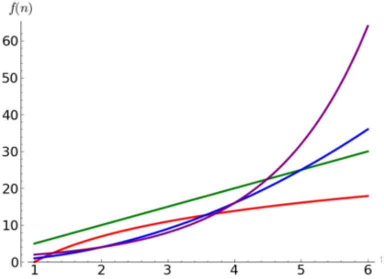
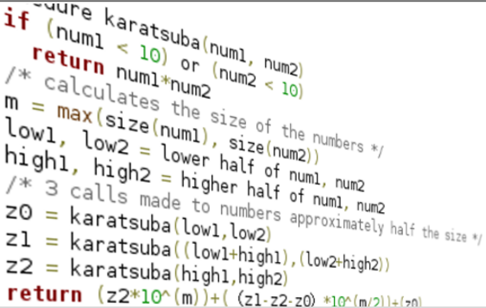
Tutorium Algorithmen 1

Simon Bischof (simon.bischof2@student.kit.edu) | 8. Juli 2013

Institut für theoretische Informatik, Prof. Sanders





Werbeblock

- 12.07.: Info-Fakultätsfest (s. algo2.iti.kit.edu/documents/algo1-2013/Vorlesungswerbung_TDI13.ppt)
- 12.07.: Mathe-Sommerfest (s. http://www.math.kit.edu/event/sommerfest/)

Lineare Programme

- n Variablen, m Constraints
- minimiere/maximiere f(x)=$\sum c_i x_i$
- $\sum a_{ij} x_i \leq b_j$ für $j \in \{1, \ldots, m\}$
- Matrixdarstellung: min/max $c^t x$ und $Ax \leq b$
- in Polyzeit lösbar
- Dualitätssatz

Lineare Programme

- n Variablen, m Constraints
- minimiere/maximiere f(x)=$\sum c_i x_i$
- $\sum a_{ij} x_i \leq b_j$ für $j \in \{1, \ldots, m\}$
- Matrixdarstellung: min/max $c^t x$ und $Ax \leq b$
- in Polyzeit lösbar
- Dualitätssatz

Lineare Programme

- n Variablen, m Constraints
- minimiere/maximiere f(x)=$\sum c_i x_i$
- $\sum a_{ij} x_i \leq b_j$ für $j \in \{1, \ldots, m\}$
- Matrixdarstellung: min/max $c^t x$ und $Ax \leq b$
- in Polyzeit lösbar
- Dualitätssatz

Lineare Programme

- n Variablen, m Constraints
- minimiere/maximiere f(x)=$\sum c_i x_i$
- $\sum a_{ij} x_i \leq b_j$ für $j \in \{1, \ldots, m\}$
- Matrixdarstellung: min/max $c^t x$ und $Ax \leq b$
- in Polyzeit lösbar
- Dualitätssatz

Lineare Programme

- n Variablen, m Constraints
- minimiere/maximiere f(x)=$\sum c_i x_i$
- $\sum a_{ij} x_i \leq b_j$ für $j \in \{1, \ldots, m\}$
- Matrixdarstellung: min/max $c^t x$ und $Ax \leq b$
- in Polyzeit lösbar
- Dualitätssatz

(Mixed) Integer Lineare Programme

- Lineare Programme mit (teilweise) ganzzahligen Variablen
- NP-schwer
- Relaxierung möglich (danach runden!)

(Mixed) Integer Lineare Programme

Karlsruher Institut für Technologie

- Lineare Programme mit (teilweise) ganzzahligen Variablen
- NP-schwer
- Relaxierung möglich (danach runden!)

(Mixed) Integer Lineare Programme

- Lineare Programme mit (teilweise) ganzzahligen Variablen
- NP-schwer
- Relaxierung möglich (danach runden!)

Greedy-Algorithmen

- treffe jeweils eine lokal optimale Entscheidung
- dadurch eventuell nicht global optimal

Greedy-Algorithmen

- Nur wenige optimale Greedy-Algorithmen:

Greedy-Algorithmen

- Nur wenige optimale Greedy-Algorithmen: Dijkstra, Jarník-Prim, Kruskal

Greedy-Algorithmen

- Nur wenige optimale Greedy-Algorithmen: Dijkstra, Jarník-Prim, Kruskal
- Oft aber Näherungslösungen (z.B. Rucksackproblem, Matchings)

Dynamische Programmierung

Anwendbar, wenn das Optimalitätsprinzip gilt:

- Optimale Lösungen bestehen aus optimalen Löungen für Teilprobleme
- Mehrere optimale Lösungen: es ist egal welche benutzt wird

Wie wendet man DP an?

- Was sind die Teilprobleme? Kreativität!
- Wie setzen sich optimale Lösungen aus Teilproblemlösungen zusammen? Beweisnot
- Bottom-up Aufbau der Lösungstabelle: einfach
- Rekonstruktion der Lösung: einfach
- Verfeinerungen (Platz sparen, Cache-effizient, Parallelisierung): Standard-Trickkiste

Aufgabe

Gegeben sei ein Farbbild, das aus einem Feld A[1..m,1..n] von Pixeln besteht. Nehmen Sie an, wir würden das Bild gerne leicht komprimieren. Genauer: Wir würden gerne ein Pixel in jeder der m Zeilen entfernen, sodass das Gesamtbild um ein Pixel schmaler wird. Um störende visuelle Effekte zu vermeiden sei gefordert, dass die entfernten Pixel in zwei benachbarten Zeilen in der gleichen oder in adjazenten Spalten liegen. Die zu entfernenden Pixel bilden eine sogenannte Naht, die in der obersten Zeile beginnt und in der untersten Zeile endet, wobei zwei aufeinanderfolgende Pixel der Naht vertikal oder diagonal zueinandern adjazent sind.

- Zeigen Sie, dass für n>1 die Anzahl der möglichen Pfade mindestens exponentiell in m wächst.

Aufgabe (Fortsetzung)

- Setzen Sie nun voraus, dass zu jedem Pixel A[i,j] ein reelwertiger Bruchwert d[i,j] existiert, der angibt, wie störend es wäre, Pixel A[i,j] zu entfernen. Intuitiv gesehen gilt, dass je kleiner der Bruchwert eines Pixels ist, umso ähnlicher ist der Pixel zu seinen Nachbarn. Setzen Sie weiterhin voraus, dass wir den Bruchwert einer Naht als die Summe der Bruchwerte ihrer Pixel definieren. Geben Sie einen Algorithmus an, um eine Naht mit einem minimalen Bruchwert zu berechnen. Geben Sie die Komplexität Ihres Algorithmus an.

Weitere Beispiele

- Edit distance/approx. string matching
- Verkettete Matrixmultiplikation
- Rucksackproblem
- Geld wechseln

Weitere Lösungsansätze

- Systematische Suche (evtl. mit Branch and Bound): z.B. Alpha-Beta-Algorithmus
- Lokale Suche: Hill Climbing
- Simulated Annealing: Genaueres in Eingebettete Systeme 2
- Evolutionäre / Genetische Algorithmen

Weitere Lösungsansätze

- Systematische Suche (evtl. mit Branch and Bound): z.B. Alpha-Beta-Algorithmus
- Lokale Suche: Hill Climbing
- Simulated Annealing: Genaueres in Eingebettete Systeme 2
- Evolutionäre / Genetische Algorithmen

Weitere Lösungsansätze

- Systematische Suche (evtl. mit Branch and Bound): z.B. Alpha-Beta-Algorithmus
- Lokale Suche: Hill Climbing
- Simulated Annealing: Genaueres in Eingebettete Systeme 2
- Evolutionäre / Genetische Algorithmen

Weitere Lösungsansätze

- Systematische Suche (evtl. mit Branch and Bound): z.B. Alpha-Beta-Algorithmus
- Lokale Suche: Hill Climbing
- Simulated Annealing: Genaueres in Eingebettete Systeme 2
- Evolutionäre / Genetische Algorithmen