Perspektiverende Datalogikursus

Perspektiverende Datalogikursus

Uge 1 - Algoritmer og kompleksitet

Gerth Stølting Brodal

27. august 2004

Indhold

Mere om

⇒ • **Eksempler på beregningsproblemer**

- Algoritmer og deres analyse
  - Korrekthed af algoritmer
  - Ressourceforbrug for algoritmer
- Kompleksitet af beregningsproblemer

Perspektiverende Datalogikursus

Beregningsproblemer

Eksempler:

- Sortering
- Søgning
- Grafer
- Strenge
- Kombinatorisk optimering
- Geometri
- Numeriske beregninger

Sortering

Problem: For elementer med en ordning tilknyttet: stil i orden.

Måske det mest fundamentale problem af alle.

Data er meget bekvemmere hvis de er sorterede. Specielt er det nemmere at lede i dem (ordbøger, telefonbøger, eksamensopslag, . . .).

Brugt som rutine i mange andre algoritmer.

Meget velstuderet problem.

Mange algoritmer (jvf. øvelserne + QuickSort + ShellSort + . . .).

Søgning

Problem: Gem data så de kan findes igen effektivt.

- Find(x)
- Insert(x)
- Delete(x)
- Successor(x), RangeSearch(x)

Et andet meget fundamentalt problem (jvf. databaser).

Brugt som rutine i mange andre algoritmer.

Meget velstuderet, mange algoritmer. To grundgrupper: søgetræer og hashing.

Grafer

Definition:

Knuder (punkter) og kanter (streger mellem punkter).

Ekstra struktur: orientering af kanter, vægte på kanter.

En *meget* anvendelig model:

> Flyruter, veje, el/vand/computer netværk,
> bekendtskaber og andre relationer,. . .

Problemer på grafer

- Løb grafen igennem (besøge alle knuder).
- Sammenhæng, k-sammenhæng.
- (Mindste) udspændende træ.
- Korteste veje.
- Euler tur.
- Hamilton tur, rejsende sælger.
- Graffarvning.
- Klike

  ⋮

Strenge

Alfabet = mængde af tegn som kan bruges

Streng = sekvens af tegn fra alfabetet

Eksempler:

“To be or not to be”

ACCCATTCCGTAA

10001100110001110

Problemer på strenge

- Pattern matching
- Regulære udtryk
- Afstandsmål (Hamming, edit)
- Længste fælles delstreng
- Længste fælles delsekvens

⋮

Kombinatorisk optimering

- Bin Packing (1D, 2D, 3D)
- Knapsack
- Subset Sum
- Job Scheduling
- Crew assignment

  ⋮

Perspektiverende Datalogikursus

Geometri

- Convex Hull
- Nearest Neighbor
- Ortogonal Line Segment Intersection
- 2D Range Search

$\vdots$

Numeriske beregninger

- Polynomieevaluering
- Matrixmultiplikation
- Løsning af ligningssystemer
- Løsning af differentialligninger

$\vdots$

Indhold

Mere om

- Eksempler på beregningsproblemer
- Algoritmer og deres analyse
  - $\Rightarrow$ **Korrekthed af algoritmer**
  - Ressourceforbrug for algoritmer
- Kompleksitet af beregningsproblemer

Invarianter

Udsagn I som gælder efter alle skridt i algoritmen.

Vælges så:

- Man kan vise at I gælder ved starten
- Man kan vise at hvis I gælder før et skridt, så gælder det efter.
- Man kan vise af I samt omstændigheder ved algoritmens afslutning implicerer det ønskede slutresultat.

Eksempler: RadixSort, binær søgning.

Indhold

Mere om

- Eksempler på beregningsproblemer
- Algoritmer og deres analyse
  - Korrekthed af algoritmer
  - ⇒ **Ressourceforbrug for algoritmer**
- Kompleksitet af beregningsproblemer

Ressourceforbrug, målestok

- RAM-modellen
- Tidsmåling vs. analyse.
- Voksehastighed, asymptotisk notation.
- Worst case, best case, average case

Først vælge (eller udvikle) algoritme efter forskelle i asymptotisk ressourceforbrug

Ved lighed, vælg dernæst efter konstanter (tidsmåling nu relevant).

Indhold

Mere om

- Eksempler på beregningsproblemer
- Algoritmer og deres analyse
  - Korrekthed af algoritmer
  - Ressourceforbrug for algoritmer
- $\Rightarrow$ **Kompleksitet af beregningsproblemer**

Perspektiverende Datalogikursus

Nedre grænser

Beviser for at ingen algoritme (blandt en stor klasse af algoritmer) kan løse problemet bedre end angivet.

Eksempler:

- Sortering
- Søgning
- Selection (find element af given rang).

Øvre og nedre grænser ens

$\Downarrow$

problemets kompleksitet kendt.

P vs. EXP

Meget grov inddeling af algoritmer i gode og dårlige:

Polynomiel tids algoritmer
kontra
Eksponentiel tids algoritmer

Eksempel: sortering vs. brute-force løsning af puslespil

P = problemer med polynomiel tids algoritme

EXP = problemer med eksponentiel tids algoritme

Klart at P $\subseteq$ EXP.

NP-fuldstændighed

NP = ja/nej-problemer, hvor en ja-løsning kan kontrolleres (men ikke nødvendigvis findes) i polynomiel tid.

Eksempel: Hamiltontur.

Det er nemt at se at P $\subseteq$ NP $\subseteq$ EXP.

NP-fuldstændigt problem S: Hvis en løsning til S kan findes i polynomiel tid, da kan der for alle problemer i NP findes en løsning i polynomiel tid.

Cook (1972): Der findes et NP-fuldstændigt problem!

Karp m.fl. (1972–): Der findes mange.

NP-fuldstændighed

NP-fuldstændigt problem S: Hvis en løsning til S kan findes i polynomiel tid, da kan der for alle problemer i NP findes en løsning i polynomiel tid.

Karp m.fl. (1972–): Der findes mange.

Eksempler: Hamiltontur, rejsende sælger, graffarvning, klike, bin packing, knapsack, subset sum, job scheduling, crew assignment,. . .

$$\boxed{\text{Formodning: P} \subsetneq \text{NP}}$$

($\Leftrightarrow$ der findes ikke polynomiel tids algoritme for disse problemer)

Hvad gør vi så med de problemer?

Hvad gør vi så?

- Heuristisk søgning
- Algoritmer for specielle instanser (jvf. “The Eternity Puzzle”).
- Approximationsalgoritmer

Flere modeller og cost-funktioner

- Online algoritmer
- Randomiserede algoritmer
- Parallelisme
- Hukommelseshierarkier

Kort sagt

Algoritmer og kompleksitet:

- Udvikle algoritmer
- Analysere algoritmer
- Analysere problemer

Ét synspunkt (David Harel, 92):

> "Algorithmics is more than a branch of computer science. It is the core of computer science, and, in all fairness, can be said to be relevant to most of science, business, and technology."