Föreläsning
Datastrukturer (DAT036)

Nils Anders Danielsson

2012-12-11

Idag

- Frågor? Är något oklart inför tentan?
- Sammanfattning.
- Jag har inte material för en hel föreläsning: avbryt mig om ni vill fördjupa er i något.

Modeller

Vi har använt förenklade modeller av verkligheten för att analysera våra datastrukturer och algoritmer.

- Uniform kostnadsmodell.
- Logaritmisk kostnadsmodell.

Modellerna har begränsningar.

Ordonotation

Ordonotation gör det möjligt att dölja irrelevanta – men även relevanta – detaljer.

- Definition (O, Ω, Θ).
- Regler.

Tidskomplexitetsanalys

Vi har analyserat många algoritmers tidskomplexitet.

En enkel analys kan se ut så här:

- Vi känner till tidskomplexiteten hos vissa standardkomponenter.
- Vi räknar sedan hur många gånger standardkomponenterna körs.
  - ”En gång per varv i loopen.”
  - ”En gång per nod.”
  - ”En gång per kant.”

Pseudokod

- Blandning av programmeringsspråk, matematisk notation och naturligt språk.
- Mål: Fokusera på det viktiga, undvik onödiga detaljer.
- Ibland kan det vara bra med fler detaljer, ibland färre.

Abstrakt datatyp/datastruktur

- Abstrakt datatyp: matematisk abstraktion.
- Datastruktur: implementation.

Några (klasser av) abstrakta datatyper

- Listor.
- Stackar.
- FIFO-köer.
- Prioritetsköer.
- Mängder.
- Avbildningar (”maps”).
- Grafer.

Tips: Återanvänd kod

- Ofta behöver man inte implementera datastrukturer själv.
- Om man någon gång behöver det kan man ändå dra nytta av existerande standarddatastrukturer.

2012/08:2

Uppgiften är att konstruera en datastruktur som representerar ”dubbelriktade maps” (bijektioner mellan ändliga mängder):
`insert(`s`,`t`)`, `target(`s`)`, `source(`t`)`.

```
public class Bijection<S, T> {

  private Map<S, T> to;
  private Map<T, S> from;

  public void insert(S s, T t) {
    if (to.containsKey(s) || from.containsKey(t)) {
      throw new IllegalArgumentException();
    }
    to.put(s, t);
    from.put(t, s);
  }

  public T target(S s) {
    return to.get(s);
  }

  ...
}
```

Listor

ADT med olika implementationer.

Länkade listor:

- Långsam indexering.
- Snabb insättning (exkl sökning) i mitten.
- Vaktposter? Enkellänkade, dubbellänkade?
- Pekarjonglering. Testa! Invarianter.

Dynamiska arrayer:

- Snabb indexering.
- Snabb insättning sist.
- Amorterad tidskomplexitet.

Sortering

- Insättningssortering: $O(n^2)$.
- Mergesort: $O(n \log n)$.
- Heapsort: $O(n \log n)$.
- Quicksort: $O(n \log n)$ (medel).
- Räknesortering: $O(N + n)$ (N: antalet hinkar).
- Radixsortering: $O(d(N + n))$
  (N: "talbasen", d: största antalet "siffror").

Stackar, FIFO-köer

- Enkla implementationer med (olika sorters) listor.
- Cirkulära arrayer.

Prioritetsköer

- Binära heapar:
  - Heapordnade kompletta binära träd.
  - Trädet representeras ofta av en array.
  - Bubbling.
- Binomialheapar:
  - Baserade på positionellt talsystem: $5 = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2$.
  - Amorterad tidskomplexitet.

Mängder, avbildningar

Hashtabeller:

- Hashfunktioner.
- Kollisioner.
- Kedjning, öppen adressering.

Mängder, avbildningar

Olika sorters sökträd:

- Obalanserade binära sökträd:
  - Långsamma vid insättning av sorterad sekvens.
- AVL-träd:
  - Höjdbalanserade.
  - Rotationer.
- Splayträd:
  - Ej balanserade, ”splayning”.
  - Amorterad tidskomplexitet.
- Rödsvarta träd, 2-3-träd, B^+-träd, ...

Mängder, avbildningar

Skipplistor:

- Insättning: randomiserad algoritm.
- Förväntad tidskomplexitet.

Grafer

- ADT: $G = (V, E)$.
- Datastrukturer: grannlistor, grannmatriser.
- Algoritmer:
  - Topologisk sortering.
  - Kortaste vägen: bredden först-sökning, Dijkstra, Floyd-Warshall.
  - Minsta uppspännande träd: Prim, Kruskal (inkl disjunkt mängd-datastruktur).
  - Djupet först-sökning.
  - Starkt sammanhängande komponenter: Kosaraju.

Till sist

- Sista övningstillfället: gamla tentauppgifter.
- Svara gärna på kursenkäten.
  Tycker ni t ex att vi ska ha en dugga nästa år?

Lycka till!