Institut for Matematik og Datalogi
Syddansk Universitet, Odense

31. marts 2016
RF

DM507 – Opgaver uge 14

Eksaminatorier I

1. Eksamen juni 2009, opgave 1, spørgsmål b.

2. (∗) Cormen et al. øvelse 14.2-4 (side 348). Opgaven kan løses uden at læse kapitel 14. Operationen kaldes langt oftere RangeSearch end Enumerate. Hint: lade dig inspirere af Inorder-Tree-Walk (side 288). Dette giver ret nemt algoritmen, og det udfordrende i opgaven er så at finde et argument for køretiden.

3. Cormen et al. øvelse 14.1-1 (side 344).

4. Cormen et al. øvelse 14.1-2 (side 344).

5. (∗) Cormen et al. øvelse 14.1-5 (side 344). At læse "find" i stedet for "determine" gør måske opgaven lidt klarere.

6. (∗) Cormen et al. øvelse 14.1-7 (side 345). Hint: tænk på insertionsort, men indsæt i et træ.

7. Cormen et al. øvelse 11.2-2 (side 261).

8. Cormen et al. øvelse 11.4-1 (side 277).

9. Eksamen juni 2013, opgave 5.

10. Implementer Countingsort i Java ud fra bogens pseudokode (side 171). Husk at sætte en øvre grænse k på de `int`'s, du sorterer. Test at din kode fungerer ved at generere arrays med forskelligt indhold og sortere dem. Tilføj tidtagning af din kode (kun selve sorteringen, ikke den del af programmet som genererer array'ets indhold).

    Kør derefter din kode med input, som er random `int`'s i intervallet $[0; k]$ for $k = n/50$, n, og $50n$ (dvs. tre værdier af k for hver værdi af n). Brug f.eks. `java.util.Random.nextInt(k+1)` til generering af tallene. Gør dette for mindst tre forskellige værdier af n (antal elementer at sortere), vælg værdier som får programmet til at bruge fra ca. 100 til ca. 5000 millisekunder for $k = n$ kørslen. Gentag hver enkelt kørsel tre gange og find gennemsnittet af antal millisekunder brugt ved de tre kørsler. Dividér de fremkomne tal med $n + k$ og check derved hvor godt analysen passer med praksis – de resulterende tal burde ifølge analysen være konstante.

Sammenlign med dine køretider for det tilsvarende forsøg (samme n) med Quicksort fra opgaverne i uge 11 (eller evt. Mergesort fra uge 9). Er Quicksort eller Countingsort hurtigst? Afhænger det af k (for fastholdt n)?

Eksaminatorier II

1. Eksamen januar 2008, opgave 3.
2. Eksamen januar 2008, opgave 1c.
3. Eksamen januar 2006, opgave 1a.
4. Cormen et al. øvelse 2.1-3 (side 22). Kun spørgsmålet om invariant, resten er lavet tidligere.
5. Cormen et al. øvelse 2.2-2 (side 29).
6. Cormen et al. opgave 2-2 (side 40). Hint til **c**: tænk på Selectionsort.
7. Eksamen juni 2013, opgave 6.
8. Eksamen juni 2010, opgave 3.

Studiegrupper

Forslag til fokus for arbejde i studiegrupper (hvis man er i en sådan):

- Genfortæl for hinanden ideen i rød-sorte træer: hvad er strukturkravet, hvorfor giver dette $O(\log n)$ højde, hvordan er rebalancering efter henholdsvis indsættelse og sletning bygget op, hvorfor virker det (dvs. fjerner overtrædelser af strukturkravet), og hvorfor tager det $O(\log n)$ tid?

- Diskuter hvad projektet del II går ud på, og hvordan I løser/har løst det.
- Forsøg at lave alle opgaverne på forhånd. Sammenlign svar i studiegruppen. Skiftes til at fremlægge jeres løsning. For de opgaver, hvor alle var gået i stå, forsøg at løse dem igen i fælleskab.