Institut for Matematik og Datalogi
Syddansk Universitet, Odense

8. marts 2018
RF

DM507 – Opgaver uge 11

Eksaminatorier I

1. Eksamen juni 2008, opgave 4 b.
2. Cormen et al. øvelse 12.2-1 (side 293), spørgsmål a–c.
3. Cormen et al. øvelse 12.2-3 (side 293).
4. Cormen et al. øvelse 12.3-3 (side 299). Bemærk at opgaven mener ubalancerede søgetræer (kapitel 12). Besvar bagefter opgaven igen, men nu med rød-sorte træer i stedet for ubalancerede binære søgetræer.
5. Cormen et al. øvelse 12.1-5 (side 289)
6. Cormen et al. øvelse 12.1-2 (side 289)
7. (∗) 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.
8. Cormen et al. øvelse 13.1-2 (side 311).
9. Eksamen juni 2011, opgave 1.
10. Eventuelle spørgsmål om løsning af Multiple Choice Test 1.

Eksaminatorier II

1. Eksamen jan 2005, opgave 1.
2. Eksamen juni 2009, opgave 1, spørgsmål b.
3. Cormen et al. øvelse 11.2-2 (side 261).
4. Cormen et al. øvelse 11.4-1 (side 277).

5. Eksamen januar 2008, opgave 1c.

6. Eksamen januar 2006, opgave 1a.

7. Eksamen juni 2013, opgave 5.

8. Implementer Countingsort i Java ud fra bogens pseudokode (side 195). 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 10 (eller evt. Mergesort fra uge 8). Er Quicksort eller Countingsort hurtigst? Afhænger det af k (for fastholdt n)?

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?

Forbered dele af opgaverne til eksaminatorietimer, f.eks. på nedenstående måde.

- Lav eksamensopgaverne individuelt på forhånd, og ret hinandens i gruppen.

- I opgave II.8, lav programmeringen og programkørsel (gerne i par) før gruppemøde. På mødet, sammenlign køretider.