Institut for Matematik og Datalogi
Syddansk Universitet, Odense

7. marts 2025
RF

Opgaver Uge 12

DM507/DM578/DS814/SE4-DMAD

I.A: Løses i løbet af de første øvelsestimer i uge 12

De første 45 minutter handler om nedenstående opgaver.

De næste 45 minutter er for DM507 og DS814 afsat til at komme godt i gang med projektet, del I. Tanken er, at I programmerer på projektet i timen og undervejs kan få afklaret eventuelle spørgsmål.

For DM578 og SE4-DMAD kan man i stedet se på opgaverne fra del I.B.

1. Eksamen juni 2008, opgave 4 b.
2. Cormen et al., 4. udgave, øvelse 12.2-1 (side 319), spørgsmål a–c [Cormen et al., 3. udgave: øvelse 12.2-1 (side 293), spørgsmål a–c].
3. Cormen et al., 4. udgave, øvelse 12.2-3 (side 320) [Cormen et al., 3. udgave: øvelse 12.2-3 (side 293)].
4. Cormen et al., 4. udgave, øvelse 12.1-5 (side 315) [Cormen et al., 3. udgave: øvelse 12.1-5 (side 289)].
5. Eksamen juni 2013, opgave 5.

I.B: Løses hjemme inden de næste øvelsestimer i uge 12

1. Cormen et al., 4. udgave, øvelse 12.1-2 (side 315) [Cormen et al., 3. udgave: øvelse 12.1-2 (side 289)].

2. (∗) Cormen et al., 4. udgave, øvelse 13.4-9 (side 355) [Cormen et al., 3. udgave: øvelse 14.2-4 (side 348)]. Operationen kaldes langt oftere RangeSearch end Enumerate. Hint: lav enten en variant af Inorder-Tree-Walk (Cormen et al., 4. udgave, side 314 [Cormen et al., 3. udgave: side 288]) eller brug Tree-Successor (Cormen et al., 4. udgave, side 319 [Cormen et al., 3. udgave: side 292]) gentagne gange. Dette giver ret nemt algoritmen, og det udfordrende i opgaven er så at finde et argument for køretiden.

II.A: Løses i løbet af de næste øvelsestimer i uge 12

1. Cormen et al., 4. udgave, øvelse 13.1-2 (side 334) [Cormen et al., 3. udgave: øvelse 13.1-2 (side 311)].
2. Eksamen jan 2005, opgave 1.
3. Cormen et al., 4. udgave, øvelse 13.3-2 (side 346) [Cormen et al., 3. udgave: øvelse 13.3-2 (side 322)].
4. Cormen et al., 4. udgave, øvelse 13.1-6 (side 335) [Cormen et al., 3. udgave: øvelse 13.1-6 (side 312)].
5. (∗) Cormen et al., 4. udgave, øvelse 12.3-3 (side 326) [Cormen et al., 3. udgave: 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.

II.B: Løses hjemme inden øvelsestimerne i uge 13

1. Eksamen juni 2011, opgave 1.
2. Implementer Countingsort i Java eller Python ud fra bogens pseudokode (Cormen et al., 4. udgave, side 209 [Cormen et al., 3. udgave: side 195]). Husk at sætte en øvre grænse k på de `int`'s, som du genererer som input. 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)` i Java og `random.randint(0,k)` i Python 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. For hvert af ovenstående valg af n og k: kør programmet 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 9 (eller evt. Mergesort fra uge 8). Er Quicksort eller Countingsort hurtigst? Afhænger det af k (for fastholdt n)?