Institut for Matematik og Datalogi
Syddansk Universitet, Odense

21. februar 2024
RF

Opgaver Uge 9

DM507/DM578/DS814/SE4-DMAD

I.A: Løses i løbet af de første øvelsestimer i uge 9

1. Cormen et al., 4. udgave, øvelse 7.1-1 (side 186) [Cormen et al., 3. udgave: øvelse 7.1-1 (side 173)].

2. Cormen et al., 4. udgave, øvelse 7.1-2 (side 187) [Cormen et al., 3. udgave: øvelse 7.1-2 (side 174)]. Her er q værdien som PARTITION returnerer, dvs. $(i + 1)$ i linie 8 i pseudo-koden i Cormen et al., 4. udgave, side 184 [Cormen et al., 3. udgave: side 171].

   Hint til det andet spørgsmål: Start med at lave princippet for det andet område i Cormen et al., 4. udgave, figur 7.2 (side 186) [Cormen et al., 3. udgave: figur 7.2 (side 173)] om fra "$> x$"til "$\geq x$"(overvej hvorfor dette ikke vil ændre på korrektheden af selve Quicksort), og fyld så de to områder så balanceret som muligt under udførslen af PARTITION.

3. Cormen et al., 4. udgave, øvelse 7.2-2 (side 191) [Cormen et al., 3. udgave: øvelse 7.2-2 (side 178)]. Antag, at det er PARTITION fra Cormen et al., 4. udgave, side 184 [Cormen et al., 3. udgave: side 171], der bruges (ikke f.eks. PARTITION varianten lavet under andet spørgsmål i øvelse 7.1-2 ovenfor).

   Hint: i første spørgsmål i øvelse 7.1-2, som er løst ovenfor, er det undersøgt i detaljer hvad PARTITION fra Cormen et al., 4. udgave, side 184 [Cormen et al., 3. udgave: side 171] gør på det pågældende input. Argumenter for køretiden, når dette sker gentagne gange i Quicksort.

4. Cormen et al., 4. udgave, øvelse 7.2-3 (side 191) [Cormen et al., 3. udgave: øvelse 7.2-3 (side 178)]. Vis at det faktisk gælder både når input er stigende sorteret og når det er aftagende sorteret. Start med at vise det for stigende sorteret.

   Hint: undersøg i detaljer hvad PARTITION fra Cormen et al., 4. udgave, side 184 [Cormen et al., 3. udgave: side 171] gør på de pågældende input. Argumenter så for køretiden, når dette sker gentagne gange i Quicksort.

5. Eksamen juni 2008, opgave 1 b. Man må gerne referere til oplysninger i bogen, når man giver begrundelser. Quicksort antages at bruge PARTITION fra Cormen et al., 4. udgave, side 184 [Cormen et al., 3. udgave: side 171].

6. Cormen et al., 4. udgave, problem 7-2, spørgsmål b (side 200) [Cormen et al., 3. udgave: problem 7-2, spørgsmål b (side 186)].

   Svar derefter igen på Cormen et al., 4. udgave, øvelse 7.2-2 (side 191) [Cormen et al., 3. udgave: øvelse 7.2-2 (side 178)] og Cormen et al., 4. udgave, øvelse 7.2-3 (side 191) [Cormen et al., 3. udgave: øvelse 7.2-3 (side 178)], under antagelse af at Quicksort nu bruger den nye PARTITION procedure (og naturligvis laver sine to rekursive kald på de to dele af array A hvor PARTITION har placeret elementer forskellige fra $A[q]$).

I.B: Løses hjemme inden de næste øvelsestimer i uge 9

1. Implementer Quicksort i Java eller Python ud fra pseudokoden i Cormen et al., 4. udgave, side 183–184 [Cormen et al., 3. udgave: side 171]. Test at din kode fungerer ved at generere arrays med forskelligt indhold og sortere dem. Tilføj tidstagning 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. Gør dette for mindst 5 forskellige værdier af n (antal elementer at sortere), vælg værdier som får programmet til at bruge fra ca. 100 til ca. 5000 millisekunder. Gentag hver enkelt kørsel tre gange og find gennemsnittet af antal millisekunder brugt ved de tre kørsler. Divider de fremkomne tal

med $n \log_2 n$ 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 med Mergesort fra opgaverne i uge 8. Er Quicksort eller Mergesort hurtigst?

[Ekstraopgave: prøv en let optimeret variant af Quicksort, hvor pivotelementet x i PARTITION vælges ved at se på de tre elementer på første, midterste og sidste plads i den del af array'et, som skal partitioneres. Disse tre elementer sammenlignes indbyrdes, og det ordningsmæssigt midterste (medianen) af disse tre bruges som pivot-element. Gør dette for kald til PARTITION, hvor der er 16 elementer eller mere, men ikke for kald på mindre instanser (her bruges stadig bogens PARTITION). Kører denne version af Quicksort (lidt) hurtigere end standardversionen?]

I Java, gentag derefter eksperimenterne med Java's sorteringsmetode `sort` fra klassen `java.util.Arrays` (en endnu mere optimeret Quicksort implementation), og sammenlign køretiderne med dine egne implementationer af Quicksort og Mergesort. I Python, gør det samme med `sort()` metoden fra lister (denne bruger Powersort, der er en variant af Mergesort, som er designet til at udnytte eventuel eksisterende orden i input).

II.A: Løses i løbet af de næste øvelsestimer i uge 9

1. Cormen et al., 4. udgave, øvelse 6.1-6 (side 164) [Cormen et al., 3. udgave: øvelse 6.1-5 (side 154)].

2. Cormen et al., 4. udgave, øvelse 6.1-7 (side 164) [Cormen et al., 3. udgave: øvelse 6.1-6 (side 154)].

3. Eksamen januar 2008, opgave 1 b (sidehenvisningen skal være til Cormen et al., 4. udgave, side 176 [Cormen et al., 3. udgave: side 164]). Hob er en fordanskning af ordet heap.

4. Eksamen januar 2006, opgave 1 b. Bemærk at der er tale om en minheap.

5. Cormen et al., 4. udgave, øvelse 6.2-1 (side 166) [Cormen et al., 3. udgave: øvelse 6.2-1 (side 156)].

6. Udfør HEAP-EXTRACT-MAX(A) på nedenstående max-heap A.

	1	2	3	4	5	6	7	8	9	10
A:	21	18	10	12	8	9	4	7	5	2

7. Cormen et al., 4. udgave, øvelse 6.1-4 (side 164) [Cormen et al., 3. udgave: øvelse 6.1-4 (side 154)].

II.B: Løses hjemme inden øvelsestimerne i uge 10

1. Eksamen juni 2008, opgave 4 a. Hob er en fordanskning af ordet heap.
2. Cormen et al., 4. udgave, øvelse 6.5-11 (side 178) [Cormen et al., 3. udgave: øvelse 6.5-9 (side 166)].
3. ($*$) Cormen et al., 4. udgave, problem 6.2 (side 179) [Cormen et al., 3. udgave: problem 6.2 (side 167)]. For spørgsmål **b**, brug f.eks. formel (A.5) side 1147 samt inspiration fra analyse på slides af højden i binære heaps.