Institut for Matematik og Datalogi
Syddansk Universitet, Odense

6. februar 2021
RF

Opgaver Uge 6

DM507/DS814/T510040101

Husk at i DM507/DS814, samt den del af T510040101 som handler om algoritmer og datastrukturer (Rolfs del), er opgaverne til øvelsestimerne (også kaldet eksaminatorier eller e-timer) delt i to grupper:

A. En første gruppe opgaver, som man løser *i øvelsestimerne* i grupper (under Zoom f.eks. i breakout rooms), med instruktoren til rådighed for spørgsmål undervejs og med fælles opsamling til sidst. Disse opgaver skal altså *ikke* løses på forhånd. Man skal blot have læst på stoffet fra forelæsningen.

B. En anden gruppe opgaver med samme type stof. Disse løber man kort igennem til sidst i øvelsestimen, men opgaverne skal løses *hjemme* (gerne sammen med sin studiegruppe) inden de *næste* øvelsestimer. Disse opgaver er der så kort opsamling på i starten af denne næste øvelsestime.

På ugesedlerne vil lidt svære opgaver vil være mærket med ($*$), og mere svære opgaver vil være mærket med ($**$).

I denne uge er der lidt mere matematik i opgaverne til del II end normalt i kurset, hvilket skyldes karakteren af emnet (asymptotisk analyse af voksehastighed).

Erfaringsmæssigt varierer deltagernes fortrolighed med programmering en del. Hvis man ikke føler sig alt for stærk på det område, er det *netop* vigtigt at bruge tid på de programmeringsopgaver, som stilles. De er ikke mange eller store, men vil være god opvarmning til projektet. Har man god erfaring med programmering, kan man i stedet lægge en større indsats i de andre opgaver.

I.A: Løses i løbet af de første øvelsestimer i uge 6

1. Cormen et al. øvelse 2.2-3 (side 29). Svar også for best-case køretid.

2. Cormen et al. øvelse 2.1-1 (side 22). Du skal blot udføre Insertionsort i hånden, og behøver ikke lave tegningerne 100% som i bogen.

3. Implementer InsertionSort i Java eller Python ud fra pseudo-koden side 18 i lærebogen. Test at din kode fungerer ved at generere arrays/lister med forskelligt indhold og sortere dem. NB: Bogens pseudokode indekserer arrays startende med index 1, mens Java og Python starter med index 0. Man må derfor ændre passende (dvs. nogle gange bruge et index som er én mindre) i de linier i pseudokoden, som involverer indekser.

4. ($*$) Cormen et al. opgave 2-4 (side 41), spørgsmål **a**, **b** og **c**.

   Hints til spørgsmål **c**: For det oprindelige input $A[1..n]$, lad INV betegne det samlede antal inversioner og lad d_j betegne antal elementer i $A[1..(j-1)]$ som er skarpt større end $A[j]$. Lad t_j være defineret som i Cormen et al. nederst på side 25. Vis at $d_j = t_j - 1$, at $\sum_{j=2}^{n} d_j = \text{INV}$, og slut deraf via formlen midt på side 26 i Cormen et al. at køretiden for InsertionSort er $O(n + \text{INV})$

I.B: Løses hjemme inden de næste øvelsestimer i uge 6

1. Fortsæt opgaven ovenfor med implementation af InsertionSort på denne måde:

   Tilføj tidtagning af din kode ved at indsættte to kald til metoden `System.currentTimeMillis()` for Java eller `time.time()` for Python. Indsæt ét i starten af InsertionSort og ét i slutningen (slå funktionaliteten af metoden op i Javas/Pythons online dokumentation og se kodeeksemplerne fra forelæsningne i denne uge). Der skal kun tages tid på selve sorteringen, ikke den del af programmet som genererer array'ets/listens indhold.

   Kør din kode dels med sorteret input (best case for InsertionSort), dels med omvendt sorteret input (worst case for InsertionSort). Gør dette for mindst 5 forskellige værdier af n (antal elementer at sortere). Vælg disse værdier af n så de får programmet til at bruge fra ca. 100 til ca. 5000 millisekunder (værdierne er ikke de samme for best case og worst case). Gentag hver enkelt kørsel tre gange og find gennemsnittet af antal millisekunder brugt ved de tre kørsler (fluktuationer fra baggrundsprocesser får derved mindre indflydelse). Dividér de fremkomne tal med henholdsvis n (for best case input) og n^2 (for worst case input), og check derved hvor godt analysen passer med praksis – de resulterende tal burde ifølge analysen være konstante (for best case tallene og for worst case tallene, hver for sig), jvf. graferne på slides fra forelæsningen.

Kør derefter din kode med input, som er random `int`'s. I Java, brug f.eks. et `java.util.Random` objekt og dets metode `nextInt()` til at generere dem. I Python, brug f.eks. `random.randint()` til at generere dem. Er køretiderne tættest på best case eller worst case?

2. ($*$) Cormen et al. øvelse 2.3-7 (side 39). Hint: start med at sortere tallene. Du må gerne bruge at dette kan gøres i $O(n \log n)$ tid (f.eks. med Mergesort).

II.A: Løses i løbet af de næste øvelsestimer i uge 6

1. Cormen et al. øvelse 2.3-1 (side 37).
2. Cormen et al. øvelse 2.3-2 (side 37).
3. Vis for
$$f(n) = 0.1 \cdot n^2 + 5 \cdot n + 25$$
at $f(n) = \Theta(n^2)$ og $f(n) = o(n^3)$. Hint: brug sætninger fra side 17 i slides om analyse af algoritmers køretider.
4. Vis at følgende funktioner er skrevet op efter stigende asymptotisk voksehastighed:
$$1,\ \log n,\ \sqrt{n},\ n,\ n \log n,\ n\sqrt{n},\ n^2,\ n^3,\ n^{10},\ 2^n$$
Mere præcist, vis at det for alle par $f(n), g(n)$ af naboer i listen gælder at $f(n) = o(g(n))$. Hint: brug sætninger fra side 17 (og 18 og 19) i slides om analyse af algoritmers køretider.
5. ($*$) Cormen et al. øvelse 3.1-1 (side 52). (Her skal man bruge selve definitionen af $\Theta()$ fra bog/slides, ikke sætninger side 17 fra slides.)

II.B: Løses hjemme inden øvelsestimerne i uge 7

1. Cormen et al. øvelse 2.3-5 (side 39). Udvid opgaven således: Start med at illustrere algoritmen med en tegning. Lav derefter pseudokode for algoritmen (som i opgaveteksten). Lav derefter et program i Java eller Python på basis af din pseudokode. Test til sidst korrekthed af din implementation ved at generere arrays (Java) eller lister (Python) med indholdet 1,3,5,7,.... og derefter udføre søgninger. Test med både tomme arrays/lister, lange arrays/lister, og søgning både efter tal, som er der, og efter tal, som ikke er der. Husk: lad være med at snyde (dig selv og din egen læring) ved at se på kode fra nettet eller andre steder.

2. Cormen et al. øvelse 2.3-6 (side 39).

3. Cormen et al. øvelse 3.1-4 (side 53). Hint: husk regneregler for potenser (kan genopfriskes på side 55 midt).

4. ($*$) Cormen et al. øvelse 3.2-3 (side 60). Hint: for at vise formel (3.19) side 58 kan man godt bruge Stirlings formel (3.18) side 57, sådan som bogen foreslår i tekst ved (3.19), men jeg vil foreslå at argumentere langt mere jordnært ud fra definitionen af $n!$. Husk regnereglerne for logaritmer (se side 56 nederst, specielt anden regel).