

Opgaver Uge 5

DM507/DS814/SE4-DMAD

I DM507/DS814, samt den del af SE4-DMAD som handler om algoritmer og datastrukturer (Rolf's del), er opgaverne til øvelsestimerne (også kaldet eksaminatorier eller e-timer) delt i to grupper:

- A. En første samling opgaver, som man løser *i øvelsestimerne* 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 forelæsningsen. Man må meget gerne arbejde i grupper i timerne (oplagt, hvis man er i en studiegruppe allerede), da det er en stor hjælp af tale højt med andre om løsning af opgaver.
- B. En anden samling opgaver med samme type stof som den første samling. Svaret på disse opgaver løber man kort igennem til sidst i øvelsestimen mht. mulige løsningsmetoder, 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 den næste øvelsestime.

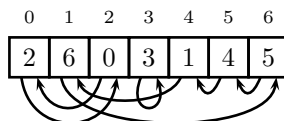
A: Løses i løbet af øvelsestimerne i uge 5

1. Cormen et al. problem 1.1 (side 14). Erstat dog “microseconds” med “nanoseconds” (dvs. 10^{-9} sekunder), da dette ca. er hvad en CPU-cyklus tager på en moderne processor. Du skal kun udfylde søjlerne *second* og *hour*, og kun rækkerne med n , $n \log n$, n^2 , n^3 og 2^n . Start

med n^2 . For nogle af indgangene kan man finde svaret ved matematisk udregning, for andre må man prøve sig frem ved at indsætte forskellige værdier af n .

2. Brodals noter om puslespil, opgave 1.
3. Brodals noter om puslespil, opgave 2. Her betyder “optimal følge af ombytninger” et antal ombytninger som angivet af sætning 1 i noterne. Opgaven viser, at andre algoritmer end “grådige algoritmer” (algoritmer som altid bringer mindst ét element på plads) kan være optimale for dette puslespilsproblem.
4. Lav et Java- eller Python-program, som genererer en tilfældig permutation af heltallene fra 0 til $n - 1$ (for et n som er en input parameter). I Java kan man bruge typen `ArrayList` samt metoden `shuffle` fra `Collections` utility klassen. I Python kan man bruge lister samt funktionen `shuffle` fra modulet `random`. Udskriv tallene i din permutation.
5. Hvis et array/en liste indeholder en permutation af tallene 0 til $n - 1$ kan man definere kredse på samme måde som for puslespillet fra første forelæsning: et tal x , som står på plads y i arrayet, giver en pil fra plads y til plads x (dvs. hvis tallet 1 står på plads 4 i arrayet, er der en pil fra plads 4 til plads 1), og en samling pile, der hænger sammen i en cyklisk kæde, kaldes en kreds.

Her er et eksempel, hvor der i alt er tre kredse i permutationen (check at du finde dem):



Lav en algoritme, som tæller antal kredse i en permutation.

Hvad er køretiden for din algoritme som funktion af n ?

Implementer din algoritme i Java eller Python.

B: Løses hjemme inden øvelsestimerne i uge 6

1. Fortsæt med Cormen et al. problem 1.1 (side 14), stadig med “microseconds” erstattet af “nanoseconds” (dvs. 10^{-9} sekunder). Du skal stadig udfylde rækkerne med n , $n \log n$, n^2 , n^3 og 2^n , men udfyld nu søjlerne *year* og *century*. For nogle af indgangene kan man finde svaret ved matematisk udregning, for andre må man prøve sig frem ved at indsætte forskellige værdier af n .
2. I øvelsestimen lavede du et program, som kunne generere en tilfældig permutation, og et program, som kunne tælle antal kredse i en permutation.

Brug disse programmer til at generere en masse tilfældige permutationer af længde 16, og tæl for hver af dem antallet af kredse i dem.

Brug data fra disse kørsler til at give et bud på sandsynligheden for, at der i en tilfældig permutation med $n = 16$ er k kredse, for $k = 1, 2, \dots, 16$. Dvs. lav din egen version af figur 3 i noterne, men med n lig 16 og ikke 64.

Find også det gennemsnitlige antal kredse i dine eksperimenter. Passer dit tal med formlen på side 4 i noterne, dvs. er det tæt på $H_{16} = \sum_{i=1}^{16} 1/i = 1/1 + 1/2 + 1/3 + \dots + 1/16$?