Institut for Matematik og Datalogi
Syddansk Universitet, Odense

27. februar 2020
RF

DM507 – Opgaver uge 10

Eksaminatorier

1. Cormen et al. øvelse 6.4-4 (side 160). Hint: Er Heapsort en sammenligningsbaseret algoritme?

2. Cormen et al. øvelse 8.2-1 (side 196).

3. Cormen et al. øvelse 8.2-3 (side 196).

4. Cormen et al. øvelse 8.2-4 (side 197).

5. Cormen et al. øvelse 8.3-1 (side 199). Brug kun de første otte ord som input (ellers bliver øvelsen for lang).

6. Eksamen juni 2008, opgave 1a.

7. Implementer Quicksort i Java eller Python ud fra bogens pseudokode (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. Dividér 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 Timsort, der er en variant af Mergesort, som er designet til at udnytte eventuel eksisterende orden i input).

8. Cormen et al. øvelse 8.3-2 (side 200). Hint til sidste del: udvid elementers nøgler.

9. (∗) Cormen et al. øvelse 8.3-4 (side 200). Hint: se på tallene som bygget op af 3 cifre $d_2d_1d_0$ i radix n. Dvs. $x = d_2 \cdot n^2 + d_1 \cdot n^1 + d_0 \cdot n^0$. Du kan i øvrigt for et tal x finde disse cifre ved at bruge heltalsdivision og modulus (rest ved heltalsdivision) med n (brug formel (3.8) side 54 gentagne gange, ligesom i algoritmen på side 16 og følgende sider på disse slides om talsystemer (hvor n er 2)).

10. (∗) Cormen et al. problem 7-4 (side 188). Hint til del c: vælg hvilken del man vil kalde rekursivt på, i stedet for at bruge den venstre del altid.

Studiegrupper

Forslag til fokus for arbejde i studiegrupper (hvis man er i en sådan):

Genfortæl for hinanden nogle af korrekthedsbeviserne fra forelæsningsslides. F.eks. hvorfor Countingsort er stabil, hvorfor Radixsort sorterer, hvorfor $\Omega(n \log n)$ er en nedre grænse for sortering ved sammenligningsbaserede algoritmer.

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

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

- I opgave 7, lav programmeringen og programkørsel (gerne i par) før gruppemøde. På mødet, sammenlign køretider.
- Forsøg at løse de mere kreative opgaver (f.eks. 1, 4, 8, 9, 10) i fællesskab. Arbejd både med at få ideer på skitseplanet til de ønskede algoritmer eller argumenter, og med at få dem formuleret præcist til sidst. I kan evt. dele dise opgaver op imellem delgrupper, som senere forsøger at formidle de fundne løsninger til hinanden så klart og præcist som muligt.
- Forsøg at lave resten af opgaverne hver især på forhånd, og sammenlign svar i studiegruppen.