Ugeseddel: Søgning og Sortering

Philip Bille

Om denne uge

Litteratur *Introduction to Algorithms*, Cormen, Rivest, Leisersons og Stein (CLRS): Kap 2.

Opgaver

1 Håndkøring og egenskaber Løs følgende opgaver.

1.1 CLRS [o] 2.1-1.

1.2 CLRS [o] 2.1-2.

1.3 CLRS 2.2-3.

1.4 CLRS [o] 2.3-1.

1.5 CLRS [C] 2.3-4.

1.6 CLRS 2.3-6.

2 Duplikater og tætte naboer Lad $A[0..n-1]$ være en tabel af heltal. Løs følgende opgaver.

2.1 [o] Et *duplikat* i A er et par af indgange i og j så $A[i] = A[j]$. Giv en algoritme der afgør om der et duplikat i A i $\Theta(n^2)$ tid.

2.2 Giv en algoritme der afgør om der et duplikat i A i $\Theta(n \log n)$ tid. *Hint:* benyt flettesortering.

2.3 Et *tætteste par* i A er et par af indgange i og j så $|A[i]-A[j]|$ er minimal blandt alle par af indgange. Giv en algoritme der finder et tætteste par i A i $\Theta(n \log n)$ tid.

3 [$D\dagger$] **Implementation af binær søgning** Implementer algoritmen til binær søgning.

4 Implementation og korrekthed af flettesortering Løs følgende opgaver.

4.1 [$\dagger$] Implementer algoritmen til fletning.

4.2 [$\dagger$] Implementer flettesortering.

4.3 [C] Vis at flettesortering sorterer enhver tabel korrekt. *Hint:* benyt induktion.

5 2Sum og 3Sum Lad $A[0..n-1]$ være en tabel af heltal (positive og negative). Tabellen A har en 2-*sum*, hvis der findes to indgange i og j, så $A[i] + A[j] = 0$. Tilsvarende, har A en 3-*sum*, hvis der findes tre indgange i, j og k så $A[i] + A[j] + A[k] = 0$. Løs følgende opgaver.

5.1 [o] Giv en algoritme, der afgør om A har en 2-sum i $\Theta(n^2)$ tid.

5.2 Giv en algoritme, der afgør om A har en 2-sum i $\Theta(n \log n)$ tid. *Hint:* benyt binær søgning.

5.3 [o] Giv en algoritme, der afgør om A har en 3-sum i $\Theta(n^3)$ tid.

5.4 Giv en algoritme, der afgør om A har en 3-sum i $\Theta(n^2 \log n)$ tid. *Hint:* benyt binær søgning.

5.5 [$**$] Giv en algoritme, der afgør om A har en 3-sum i i $\Theta(n^2)$ tid.

6 Udvælgelse, partitionering og kviksortering Lad $A[0..n-1]$ være en tabel af forskellige heltal. Tallet med *rang* k i A er det tal der fremkommer på position k såfremt man sorterer A. *Medianen* af A er tallet i A med rang $\lfloor (n-1)/2 \rfloor$. Løs følgende opgaver.

6.1 Giv en algoritme, der givet et k, finder tallet med rang k i A i $\Theta(n \log n)$ tid.

En *partitionering* af A er en opdeling af A to tabeller A_{lav} og $A_{\text{høj}}$ således at A_{lav} indeholder alle tal fra A der er mindre end eller lig med medianen af A og $A_{\text{høj}}$ indeholder alle tal fra A der er større end medianen af A. Antag i det følgende at du har en lineærtidsalgoritme til at finde medianen af en tabel.

6.2 Giv en algoritme til at beregne en partitionering af A i $\Theta(n)$ tid.

6.3 $[*]$ Giv en algoritme til at sortere A i $\Theta(n \log n)$ tid vha. rekursiv partitionering.

6.4 $[**]$ Giv en algoritme, der givet et k, finder tallet med rang k i A i $\Theta(n)$ tid.

O Obligatorisk afleveringsopgave: Manglende tal Lad A være en tabel af længde $n-1$ således at indgangene i A indeholder et unikt tal fra $\{0, 1, 2, \ldots, n-1\}$, dvs., at A indeholder alle tal fra $\{0, 1, \ldots, n-1\}$ pånær et enkelt tal m, som vi kalder det *manglende tal*. F. eks. med $A = [2, 0, 4, 3]$ ($n = 5$) er det manglende tal $m = 1$. Vi er interesserede i effektive algoritmer til at finde det manglende tal i A. Løs følgende opgaver.

O.1 Giv en algoritme, der løser problemet i $\Theta(n)$ tid. *Hint:* Brug en ekstra tabel af længde n.

O.2 Vi vil nu gerne løse problemet hurtigt, men også begrænse pladsforbruget så meget som muligt. Giv en algoritme, der løser problemet i $\Theta(n^2)$ tid og kun bruger et konstant antal ekstra variable (f. eks. 3 `int` variable i Java).

O.3 (Valgfri bonusopgave) Giv en algoritme, der løser problemet i $\Theta(n)$ tid og kun bruger et konstant antal ekstra variable.

O.4 Implementer en af dine algoritmer og beskriv kort din implementation.

O.5 Skriv et program, der afprøver din implementation på et par eksempler. Afprøv dit program og kommenter på resultaterne.