Ugeseddel: Introduktion og algoritmisk tænkning

Eva Rotenberg*

Om denne uge

Materialer Introduction to Algorithms, Cormen, Leiserson, Rivest og Stein (CLRS), kap. 1.

Opgaver

Opgaver markeret med † er implementationsopgaver, og opgaver markeret med * er særligt udfordrende.

1 Find toppunkter

Lad $A = [2, 1, 3, 7, 3, 11, 1, 5, 7, 10]$ være et array. Løs følgende opgaver:

1.1 Angiv indekserne for alle toppunkter i A.

1.2 Angiv hvilke toppunkter de to lineærtidsalgoritmer finder.

1.3 Angiv sekvensen af rekursive kald, som den rekursive algoritme producerer. Antag først at rekursionen fortsætter med venstre halvdel af arrayet hvis der er mulighed for at gå begge veje. Angiv derefter alle mulige sekvenser af rekursive kald der kan opnås ved frit valg når man kan gå begge veje.

2 Lavpunkter

Foreslå det såkaldte *lavpunktsproblem* på baggrund af toppunktsproblemet giv en præcis definition af det.

3 Algoritmer og anvendelser

1. Vælg en datastruktur som du kender fra dit indledende programmeringskursus og diskuter dens styrker og svagheder.
2. Nævn et problem fra den virkelige verden, hvor kun den optimale løsning vil være god nok. Nævn et problem fra den virkelige verden, hvor det er tilstrækkeligt at komme med en approksimering.
3. Foreslå relevante kompleksitetsmål ud over tid. Nævn mindst 3.

4 Egenskaber ved toppunkter

Lad A være et array af længde $n \geq 1$. Løs følgende opgaver:

4.1 Bevis at der altid er mindst et toppunkt i A.

4.2 Hvad er det højeste antal toppunkter, der kan være i A?

4.3 Forestil dig, at vi definerer et *supertoppunkt* som følger: Et punkt er et supertoppunkt hvis det er **skarpt** større end sine naboer (med andre ord, `A[i-1]<A[i]>A[i+1]`). Gælder de to førnævnte egenskaber også for supertoppunkter?

*baseret på Bille& Gørtz 02105

5 Toppunkter

Løs følgende opgaver:

5.1 † Implementer og afprøv en af de to lineærtidsalgoritmer til at finde toppunkter.

5.2 † Implementer den rekursive algoritme til at finde toppunkter (vær opmærksom på ikke at komme ud over grænserne på arrayet).

5.3 Beskriv hvordan det værste input til hver af de 3 toppunktsalgoritmer ser ud.

5.4 † Skriv pseudokode for en iterativ version af den rekursive algoritme til at finde toppunkter. Implementer og afprøv den.

5.5 Bevis, at den rekursive algoritme altid finder et toppunkt.

(Ledetråd: Definer en passende invariant og benyt induktion.)

6 2D toppunkter

Lad M være en $n \times n$ matrix (2D-array). Et element $M[i, j]$ er et toppunkt hvis det ikke er mindre end dets naboer i retning N, Ø, S og V (dvs. $M[i][j] \geq M[i-1][j]$, $M[i][j] \geq M[i][j-1]$, $M[i][j] \geq M[i+1][j]$ og $M[i][j] \geq M[i][j+1]$). Vi er interesseret i effektive algoritmer til at finde et toppunkt i M. Løs følgende opgaver.

6.1 Giv en algoritme til at løse problemet, og analyser dens kørselstid.

6.2 Giv en algoritme der tager $\Theta(n^2)$ tid.

6.3 [$*$] Giv en algoritme der tager $\Theta(n \log n)$ tid.

Ledetråd: Start med at finde det maksimale tal i den midterste søjle og benyt det til at lave en rekursion.

6.4 [$**$] Giv en algoritme der tager $\Theta(n)$ tid.

Ledetråd: Konstruer en rekursion der inddeler M i 4 kvadranter.