Eksaminatorie-timer

DM507: Algoritmer og datastrukturer

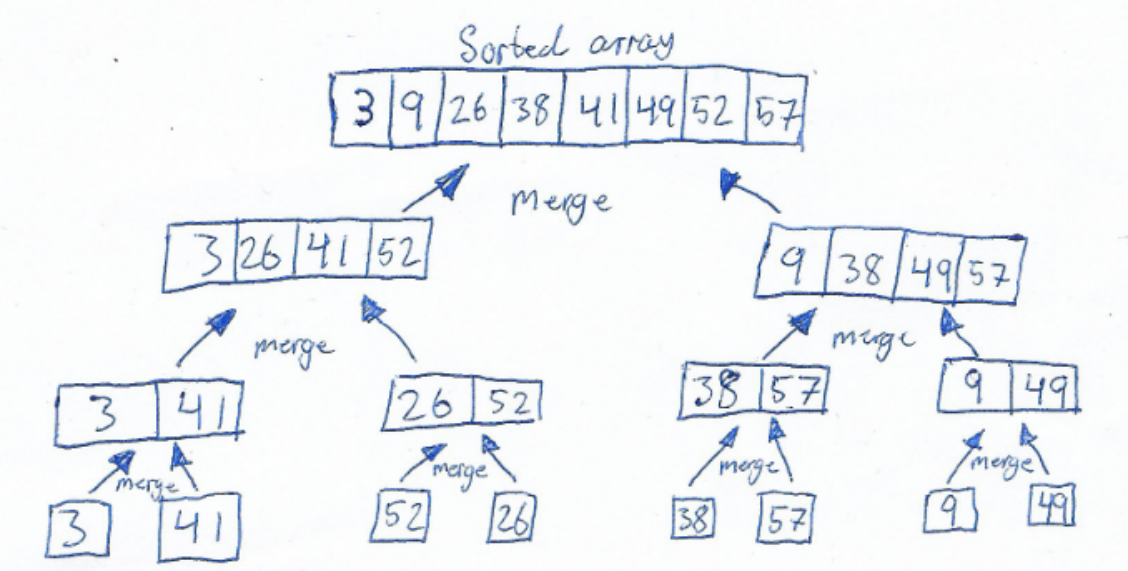
Løsninger

Uge 6 #2 2020

Del A

Opgave 1 - Cormen et al. øvelse 2.3-1

Med Figur 2.4 som model, illustrer `Merge-Sort` på arrayet $A = \langle 3, 41, 52, 26, 38, 57, 9, 49 \rangle$.



Opgave 2 - Cormen et al. øvelse 2.3-2

Omskriv `Merge` proceduren så der ikke bruges *sentinel*-værdi (f.eks. ∞), og når det ene input array er opbrugt, så kopieres resten af det andet bare direkte over i A arrayet.

```
Merge(A, p, q, r)
  n1 = q-p+1 // size of L.
  n2 = r-q // size of R.

  // Creates L and R from A.
  let L[1...n1] and R[1...n2] be new arrays
  for i=1 to n1
    L[i] = A[p+i-1]
  for i=1 to n2
    R[i] = A[q+i]

  // Actual merge part.
```

```
l = 1 // index into L.
r = 1 // index into R.
k = p // index into A.
while l <= L.length and r <= R.length:
  if L[l] <= R[r]
    A[k] = L[l]
    l = l + 1
  else
    A[k] = R[r]
    r = r + 1
  k = k + 1
// Either L or R empty, copy rest directly into A.
while l <= L.length
  A[k] = L[l]
  l = l + 1
  k = k + 1
while r <= R.length
  A[k] = R[r]
  r = r + 1
  k = k + 1
```

Opgave 3

Vis for

$$f(n) = 0.1 \cdot n^2 + 5 \cdot n + 25$$

at $f(n) = \Theta(n^2)$ og $f(n) = o(n^3)$.

Da

$$\lim_{n\to\infty} \frac{0.1 \cdot n^2 + 5 \cdot n + 25}{n^2} = \lim_{n\to\infty} 0.1 + 5 \cdot \frac{1}{n} + 25 \cdot \frac{1}{n^2} = 0.1$$

følger det af [1, p. 17], at $f(n) = \Theta(n^2)$.

Da

$$\lim_{n\to\infty} \frac{0.1 \cdot n^2 + 5 \cdot n + 25}{n^3} = \lim_{n\to\infty} 0.1 \cdot \frac{1}{n} + 5 \cdot \frac{1}{n^2} + 25 \cdot \frac{1}{n^3} = 0$$

følger det af [1, p. 17], at $f(n) = o(n^3)$.

Opgave 4

Vis at følgende funktioner er skrevet op efter stigende asymptotisk voksehastighed

$$1,\ \log n,\ \sqrt{n},\ n,\ n\log n,\ n\sqrt{n},\ n^2,\ n^3,\ n^{10},\ 2^n$$

Dette kan gøres ved at vise for alle par $f(n)$ og $g(n)$ af naboer i listen gælder at $f(n) = o(g(n))$.

(a) $1 = o(\log n)$: $\lim_{n\to\infty} \frac{1}{\log n} = 0$

(b) $\log n = o(\sqrt{n})$: $\lim_{n\to\infty} \frac{\log n}{\sqrt{n}} = \lim_{n\to\infty} \frac{\log n}{n^{\frac{1}{2}}} = 0$ (jf. [1, p. 19])

(c) $\sqrt{n} = o(n)$: $\lim_{n\to\infty} \frac{\sqrt{n}}{n} = \lim_{n\to\infty} n^{-\frac{1}{2}} = \lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0$

(d) $n = o(n \cdot \log n)$: $\lim_{n\to\infty} \frac{n}{n\cdot\log n} = \lim_{n\to\infty} \frac{1}{\log n} = 0$

(e) $n \cdot \log n = o(n \cdot \sqrt{n})$: $\lim_{n\to\infty} \frac{n \cdot \log n}{n \cdot \sqrt{n}} = \lim_{n\to\infty} \frac{\log n}{\sqrt{n}} = 0$ (se (b))

(f) $n \cdot \sqrt{n} = o(n^2)$: $\lim_{n\to\infty} \frac{n \cdot \sqrt{n}}{n^2} = \lim_{n\to\infty} \frac{\sqrt{n}}{n} = \lim_{n\to\infty} \frac{n^{\frac{1}{2}}}{n} = \lim_{n\to\infty} \frac{1}{n^{\frac{1}{2}}} = 0$

(g) $n^2 = o(n^3)$: $\lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{1}{n} = 0$

(h) $n^3 = o(n^{10})$: $\lim_{n\to\infty} \frac{n^3}{n^{10}} = \lim_{n\to\infty} \frac{1}{n^7} = 0$

(i) $n^{10} = o(2^n)$: $\lim_{n\to\infty} \frac{n^{10}}{2^n} = 0$ (jf. [1, p. 18])

Opgave 5* - Cormen et al. øvelse 3.1-1

Lad $f(n)$ og $g(n)$ være asymptotisk ikke-negative funktioner. Vha. definitionen på Θ-notation

$\Theta(g(n)) = \{f(n) | \exists$ positive konstanter c_1, c_2, n_0 s.t. $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for alle $n \geq n_0\}$

bevis at $\max(f(n), g(n)) = \Theta(f(n)) + g(n))$.

Fra definitionen skal følgende gælde:

$$0 \leq c_1 \cdot (f(n) + g(n)) \leq \max(f(n), g(n)) \leq c_2 \cdot (f(n) + g(n))$$

Hvad skal c_1 og c_2 være for uligheden er dækket?

- Hvis $c_1 = \frac{1}{2}$ er venstre ulighed dækket. Observér $f(n)+g(n)$ højst kan være $2 \cdot \max(f(n), g(n))$.
- Hvis $c_2 = 1$ er højre ulighed dækket, da højresiden altid vil være mindst $\max(f(n), g(n))$.

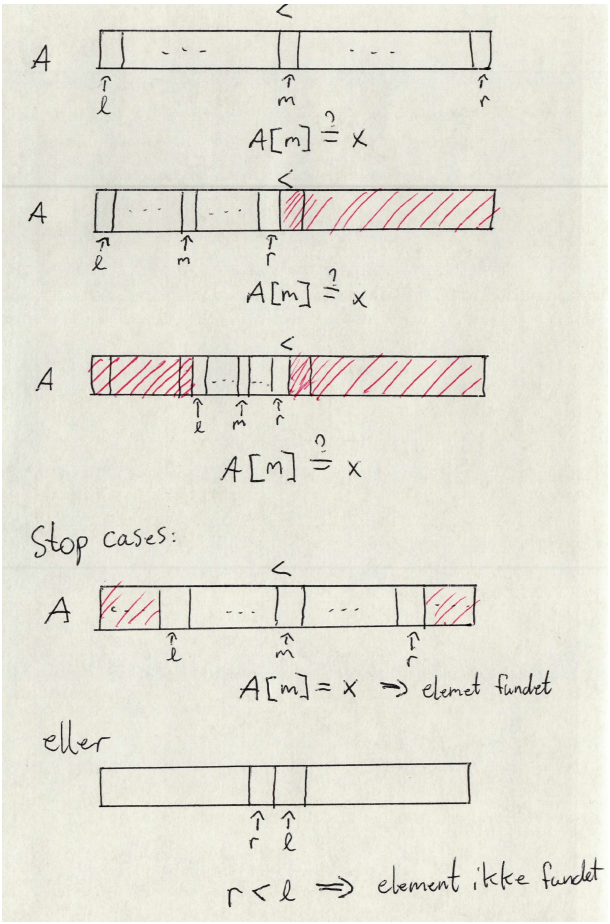
Bemærk: uden konkrete værdier for $f(n)$ og $g(n)$ kan vi ikke bestemme n_0.

Del B

Opgave 1 - Cormen et al. øvelse 2.3-5

Betragt *Søgeproblemet* igen, og observér hvis input arrayet A er sorteret, så kan vi bare tjekke midten af arrayet, og dermed udlukke halvdelen af elementerne fra den videre søgning (hvis x ikke var det midterste element). Skriv pseodokode for enten en iterativ eller rekursiv udgave af *Bineær søgning*. Argumentér for at worst-case køretiden er $\Theta(\lg n)$.

Hvad er fordelen ved A er sorteret? Vi kan udlukke halvdelen af elementerne i hver iteration fremfor f.eks. et enkelt element. I hvert case under gælder inden $A[m] = x$ tjekkes: hvis x er i A, så er x indenfor $A[l..r]$.



Følgende er pseudokode for en iterative udgave af Bineær søgning:

```
BinarySearch(A, v)
  l = 1
  r = A.length
  m = (l+r)/2 // implicit floor-funktion

  while l<=r and A[m] != v
    if v < A[m]
      r = m-1
    else
      l = m+1
    m = (l+r)/2 // implicit floor-funktion
```

```
return l<=r ? m : NIL
```

Køretids-analyse: Husk antallet af elementer halveres i hver iteration. Dvs. efter den første iteration er der $\leq \frac{n}{2}$ elementer, efter en anden iteration er der $\leq \frac{n}{4}$ elementer... Den sidste iteration sker når $\frac{n}{2^i} \geq 1$ og $\frac{n}{2^{i+1}} < 1$. Isoleres i i begge udtryk fås $\lg n - 1 < i \leq \lg n$. Heraf følger at køretiden er $\Theta(\lg n)$, da hver iteration kræver konstant arbejde.

Opgave 2 - Cormen et al. øvelse 2.3-6

Observér at while-loopet på linje 5-6 i `Insertion-Sort` bruger lineær søgning for at skanne baglæns igennem det sorterede subarray $A[1 \ldots j-1]$. Kan bineær søgning bruges i stedet, så worst-case køretiden formindskes til $\Theta(n \lg n)$.

Nej. Vi kan modificere `BinarySearch(A, v)` sådan den giver hvor et element skal placeres for at opretholde ordenen. I worst-case skal hvert element gøres til det første element på et givent tidspunkt under udførelsen, hvilket vil give en worst-case køretid på $\Theta(n^2)$, da elementerne i $A[1 \ldots j-1]$ skal rykkes, så der er plads.

Opgave 3 - Cormen et al. øvelse 3.1-4

Er $2^{n+1} = O(2^n)$? Er $2^{2n} = O(2^n)$?

Da

$$\lim_{n\to\infty} \frac{2^{n+1}}{2^n} = \lim_{n\to\infty} 2 = 2$$

følger det af [1, p. 16-17] at $2^{n+1} = \Theta(2^n) \;\Rightarrow\; 2^{n+1} = O(2^n)$.

Da

$$\lim_{n\to\infty} \frac{2^n}{2^{2n}} = \lim_{n\to\infty} \frac{2^n}{2^n \cdot 2^n} = \lim_{n\to\infty} \frac{1}{2^n} = 0$$

følger det af [1, p. 16-17] at $2^n = o(2^{2n}) \;\Leftrightarrow\; 2^{2n} = \omega(2^n)$. Heraf følger at $2^n = O(2^{2n})$ ikke er muligt, da $2^{2n} = \omega(2^n)$.

Opgave 5* - Cormen et al. øvelse 3.2-3

- Bevis $n! = \omega(2^n)$

  Da

  $$\lim_{n\to\infty} \frac{2^n}{n!} = \lim_{n\to\infty} \underbrace{\frac{2}{n} \cdot \frac{2}{n-1} \cdots \frac{2}{2} \cdot \frac{2}{1}}_{\text{alle faktorer } \leq 2} \overset{(*)}{=} 0$$

  følger det af [1, p. 17] at $2^n = o(n!) \;\Leftrightarrow\; n! = \omega(2^n)$.

  (*): Grænseværdien kan tages for hver faktor, da de alle konvergerer, men eftersom minimum én (f.eks. $\lim_{n\to\infty} \frac{2}{n} = 0$) konvergerer mod 0, så er produktet af grænseværdierne 0.

- Bevis $n! = o(n^n)$

  Da

$$\lim_{n\to\infty} \frac{n!}{n^n} = \lim_{n\to\infty} \underbrace{\frac{n}{n} \cdot \frac{n-1}{n} \cdots \frac{2}{n} \cdot \frac{1}{n}}_{\text{alle faktorer } \leq 1} \overset{(*)}{=} 0$$

  følger det af [1, p. 17] at $n! = o(n^n)$.

  (*): Grænseværdien kan tages for hver faktor, da de alle konvergerer, men eftersom minimum én (f.eks. $\lim_{n\to\infty} \frac{1}{n} = 0$) konvergerer mod 0, så er produktet af grænseværdierne 0.

- Bevis $\lg(n!) = \Theta(n \lg n)$

  Jf. [1, p. 16] har vi

$$f(n) = O(g(n)) \ \wedge \ f(n) = \Omega(g(n)) \ \Rightarrow \ f(n) = \Theta(g(n))$$

  $\underline{\lg(n!) = O(n \lg n)}$: Da

$$\begin{aligned} \lg(n!) &= \lg n + \lg(n-1) + \cdots + \lg 2 + \lg 1 \\ &\overset{(*)}{\leq} \lg n + \lg n + \cdots + \lg n + \lg n \\ &= c \cdot n \lg n, \quad c = 1 \end{aligned}$$

  følger det af definitionen af O-notation [1, p. 11], at $\lg(n!) = O(n \lg n)$.

  (*): Hvert led på venstre side er $\leq$ det tilsvarende led på højre side.

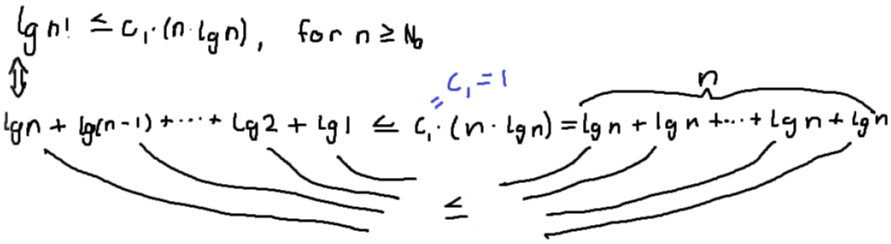


Figure 1: Sammenhørende led brugt til at vise $\lg(n!) = O(n \lg n)$.

  $\underline{\lg(n!) = \Omega(n \lg n)}$: Da

$$\begin{aligned} \lg(n!) &= \lg n + \lg(n-1) + \cdots + \lg(\tfrac{n}{2}) + \cdots + \lg 2 + \lg 1 \\ &\geq \lg n + \lg(n-1) + \cdots + \lg(\tfrac{n}{2}) \\ &\geq \lg(\tfrac{n}{2}) + \lg(\tfrac{n}{2}) + \cdots + \lg(\tfrac{n}{2}) \\ &= \frac{n}{2} \cdot \lg(\tfrac{n}{2}) \\ &\overset{(**)}{\geq} \frac{n}{2} \cdot (\frac{1}{2} \lg n) \\ &= c \cdot n \cdot \lg n, \quad c = \frac{1}{4} \end{aligned}$$

følger det af definitionen af Ω-notation [1, p. 12], at $\lg(n!) = \Omega(n \lg n)$.

(**): For $n \geq 4$ gælder: $\lg(\frac{n}{2}) = \lg n - \lg 2 = \underbrace{\lg n - 1 \geq \frac{1}{2} \cdot \lg n}_{\text{her bruges at } n \geq 4}$

Da $\lg(n!) = \Omega(n \lg n)$ og $\lg(n!) = O(n \lg n)$ følger det at $\lg(n!) = \Theta(n \lg n)$.

References

[1] Rolf Fagerberg. Asymptotisk analyse af algoritmers køretider. URL `https://imada.sdu.dk/~rolf/Edu/DM507/F21/asymptotiskAnalyseAfAlg.pdf`, 2021.