Eksaminatorie-timer

DM507: Algoritmer og datastrukturer

Løsninger

Uge 7 2021

Del A

Opgave 1 - Eksamen juni 2008, opgave 2

Angiv den asymptotiske rækkefølge af følgende funktioner:

$$\sqrt{n},\ \ 2^n,\ \ (\log_{10} n)^2,\ \ n,\ \ \log_2 n$$

Ud fra intuition antages det, at den asymptotiske rækkefølge er:

$$\log_2 n,\ \ (\log_{10} n)^2,\ \sqrt{n},\ \ n,\ \ 2^n$$

Vis nu at den postulerede rækkefølge er korrekt ved at vise for alle par $f(n)$ og $g(n)$ af naboer i listen gælder at $f(n) = o(g(n))$:

- $\log_2 n = o((\log_{10} n)^2)$: $\frac{\log_2 n}{(\log_{10} n)^2} = \frac{\frac{\log_{10} n}{\log_{10} 2}}{(\log_{10} n)^2} = \frac{\frac{1}{\log_{10} 2}}{\log_{10} n} \to 0$ når $n \to \infty$
- $(\log_{10} n)^2 = o(\sqrt{n})$: $\frac{(\log_{10} n)^2}{\sqrt{n}} = \frac{(\log_{10} n)^2}{n^{\frac{1}{2}}} \to 0$ når $n \to \infty$ (jf. [1, p. 19])
- $\sqrt{n} = o(n)$: $\frac{\sqrt{n}}{n} = \frac{\sqrt{n}}{\sqrt{n} \cdot \sqrt{n}} = \frac{1}{\sqrt{n}} \to 0$ når $n \to \infty$
- $n = o(2^n)$: $\frac{n}{2^n} \to 0$ når $n \to \infty$ (jf. [1, p. 18])

Opgave 2 - Eksamen juni 2011, opgave 2

Denne opgave handler om asymptotisk notation.
Lad f_1, f_2, g_1 og g_2 være positive funktioner, som opfylder, at

- $f_1(n) = O(g_1(n))$
- $f_2(n) = O(g_2(n))$

Hvilke af følgende tre udsagn er da sande?

a) $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$: Sand, da

Jf. antagelsen of definitionen af O-notation, så har vi

$$f_1(n) \leq c_1 \cdot g_1(n),\ n \geq n_1$$
$$f_2(n) \leq c_2 \cdot g_2(n),\ n \geq n_2$$

og det medfører

$$f_1(n) + f_2(n) \leq \max(c_1, c_2) \cdot (g_1(n) + g_2(n)),\ n \geq \max(n_1, n_2)$$

b) $g_1(n) + g_2(n) = \Omega(f_1(n) + f_2(n))$: Sand - følger af (a) da

$$f_1(n) + f_2(n) = O(g_1(n) + g_2(n)) \quad \Leftrightarrow \quad g_1(n) + g_2(n) = \Omega(f_1(n) + f_2(n))$$

c) $\frac{f_1(n)}{f_2(n)} = O(\frac{g_1(n)}{g_2(n)})$: Falsk, da

Lad $f_1(n) = 2^n$, $f_2(n) = 1$, $g_1(n) = 2^n$ og $g_2(n) = 2^n$.
Vi har $f_1(n) = O(g_1(n))$ og $f_2(n) = O(g_2(n))$, men

$$\frac{2^n}{1} = O(\frac{2^n}{2^n}) \quad \Leftrightarrow \quad 2^n = O(1)\,.$$

Opgave 3 - Mergesort

Se koden i mappen `week_8/mergesort`

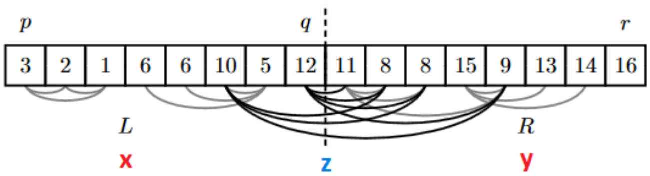
Opgave 4 - Cormen et al. opgave 2-4, spørgsmål (*)

Giv en algoritme som bestemmer antallet af inversioner i enhver permutation med n elementer i $\Theta(n \lg n)$ worst-case tid. (Hint: Modificer mergesort)

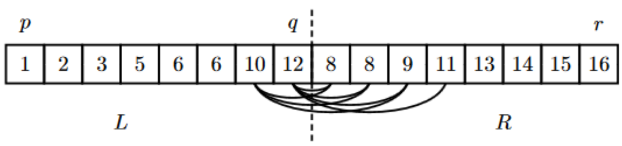
Hvordan kan man tælle inversioner? For hvert element, tæl antallet af elementer med lavere indeks[1], som den indgår i en inversion med. Læg alle disse sammen, og så har vi antallet af inversioner.

Observér hvis et (del)array $A[p..r]$ opdeles i to dele $L = A[p..q]$ og $R = A[(q+1)..r]$, så er antallet af inversioner $x + y + z$, hvor

- x er antallet af inversioner i L;
- y er antallet af inversioner i R;
- z er antallet af inversioner mellem L og R.



Derudover, hvis L og R sorteres, så vil inversionerne mellem L og R stadig være der:



[1] Jf. definition af inversion, så er der kun en inversion mellem i og j hvis $i > j$, men $A[i] < A[j]$.

God idé: når der merges (i mergesort) så rykkes elementer forbi hinanden, som måske indgår i en inversion. Hvordan kan vi tælle antallet af inversioner i merge? Husk: vi har to sorterede lister L og R, hvor det første element i R (og dermed alle andre i R) kommer efter elementerne i L. Under merge af L og R er der to cases.

- Case 1: mindste element er første i L eller lighed mellem første element i L og R; første element i L vælges. Ingen inversioner ophæves da det står rigtigt ift. de resterende elementer i L og R.

- Case 2: mindste element er første i R; første element i R vælges. Antallet af inversioner der ophæves er lig antallet af elementer tilbage i L. Hvorfor? Elementet der flyttes, flyttes kun forbi de tilbageværende elementer i L. Da alle disse har lavere indeks og større værdi (da man ellers ville være i case 1), så indgår elementet fra R i en inversion med hvert af disse elementer.

Hvordan skal mergesort modificeres: I alle kald til merge skal der altså være en counter som returneres og summeres sammen til sidst. Hver counter skal tælle antallet af ophævede inversioner under merge.
I pseudokode kunne det ligne noget à la:

```
Merge_inv(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+1] and R[1...n₂+1] be new arrays
  for i=1 to n₁
    L[i] = A[p+i-1]
  for i=1 to n₂
    R[i] = A[q+i]
  L[n₁+1] = ∞
  R[n₂+1] = ∞

  // Actual merge part.
  i = 1
  j = 1
  counter = 0
  for k = p to r
    if L[i] <= R[j]
      A[k] = L[i]
      i = i+1
    else
      A[k] = R[j]
      counter += n₁ - i + 1 // adds number of remaining elements in A
      j = j+1
  return counter
```

og

```
Mergesort_inv(A, p, r)
  if p < r
    q = ⌊(p+r)/2⌋
    x = Mergesort_inv-sort(A,p,q)
    y = Mergesort_inv
    z = Merge_inv(A,p,q,r)
    return x+y+z
```

Eksempel:

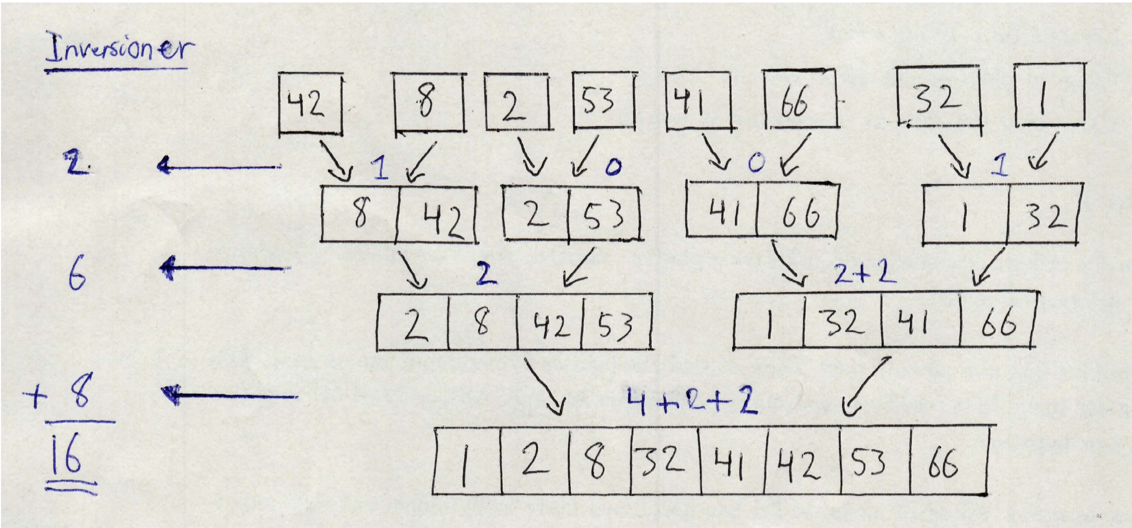


Figure 1: Udførsel af algoritme beskrevet ovenover. De blå tal er *counteren* retuneret fra kald til (modificeret) merge.

Del B

Opgave 1 - Eksamen januar 2007, opgave 3

Hvilke af følgende fem udsagn er sande?

1) $n^2 = \Omega(n)$: Sand, da $\underbrace{n = O(n^2)}_{\text{jf. uge 6}}$.

2) $n = \Theta(n^2)$: Falsk, da $\underbrace{n = o(n^2)}_{\text{jf. uge 6}}$.

3) $n \log n = o(n^2)$: Sand, da $\frac{n \log n}{n^2} = \frac{\log n}{n} \to 0$ når $n \to \infty$

4) $\log n = O(\sqrt{n})$: Sand, da $\underbrace{\log n = o(\sqrt{n})}_{\text{jf. uge 6}} \Rightarrow \log n = O(\sqrt{n})$.

5) $n! = \omega(2^n)$: Sand - dette blev vist i **Uge 6**.

Opgave 2 - Mergesort

Se koden i mappen `week_8/mergesort`

References

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