Algoritmer og Datastrukturer (NDAA04010U)
Ugeopgave 2 med svar og forklaringer

Københavns Universitet

2024

1 Amortiseret analyse

I denne opgave betragter vi potentialfunktionen der bruges til at analysere en binær tæller i CLRS sektion 16.3. Lad $\Phi(D_i)$ betegne potentialfunktionens værdi efter i Increment operationer.

a) Hvilke af nedenstående egenskaber har $\Phi(D_i)$? (Læg mærke til at udsagnene ikke er uafhængige — hvis der gælder $f(i) = \Omega(i)$ så gælder også $f(i) = \Omega(\lg i)$, osv.) Vælg ét eller flere korrekte svar og beskriv hvordan du kom frem til dem (både positive og negative svar).

1. $\Phi(D_i) = \Omega(\lg i)$.
2. $\Phi(D_i) = \Omega(i)$.
3. $\Phi(D_i) = O(1)$.
4. $\Phi(D_i) = O(\lg i)$. ✓
5. $\Phi(D_i) = O(i)$. ✓

Svar: $\Phi(D_i)$ er lig med antal 1er i den binære repræsentation af tallet i (som er mindst 1 for $i > 0$). Hvis i er en potens af 2 er $\Phi(D_i) = 1$, så vi kan ikke begrænse funktionen nedefra med en voksende funktion af i. Alle 1er i den binære repræsentation af i er blandt de $\lceil \log i \rceil + 1$ mindst betydende bits, så $\Phi(D_i) \leq \lceil \log i \rceil + 1$, dvs. både $O(\lg i)$ og $O(i)$ er gyldige øvre grænser. △

Betragt nu situationen hvor vi gerne vil analysere en tæller i det almindelige base-10 talsystem. Tælleren skal repræsentere heltal med op til k decimaler, startende med $00\ldots0$ (dvs. k nuller) og understøtte en `Increment` operation der øger tællerens værdi med 1 (op til den maksimake værdi $10^k - 1$). Lad x_i betegne værdien af tælleren efter i operationer (dvs. vi har $x_i = i$).

b) Definér en potentialfunktion $\Phi(x_i)$ og brug den til at vise, at base-10 tælleren understøtter INCREMENT i amortiseret tid $O(1)$.

Svar: Der er mange mulige potentialfunktioner, hér gives blot ét eksempel. Vi definerer $\Phi(x_i) = b(x_i)$, hvor $b(x_i)$ er antal 9ere i repræsentationen af x_i. Omkostningen c_i ved at øge tællerens værdi fra x_{i-1} til $x_i = x_{i-1} + 1$, målt i antal decimaler som ændres, er højst $t_i + 1$, hvor t_i er antallet af sammenhængende 9ere i slutningen af decimalrepræsentationen af x_{i-1}. Den amortiserede omkostning er derfor:

$$\begin{aligned}
\hat{c}_i &= c_i + (\Phi(x_i) - \Phi(x_{i-1})) \\
&\leq (t_i + 1) + (b(x_i) - b(x_{i-1})) \\
&\leq (t_i + 1) + (1 - t_i) \\
&= 2\ .
\end{aligned}$$

Den sidste ulighed følger af at decimalrepræsentationen af x_i fås fra decimalrepræsentationen af x_{i-1} ved at lave t_i 9ere om til 0er og højst introducere én ny 9er. $\triangle$

Antag nu at vi gerne også vil understøtte en `Decrement` operation der reducerer tællerens værdi med 1 (ned til den minimale værdi 0).

c) Argumentér for at base-10 tælleren understøtter `Decrement` i amortiseret tid $O(k)$.

Svar: Antal decimaler c_i der skal ændres ved en DECREMENT operation er højst k. Vi har at $\Phi(x_i) \in [0, k]$ så den amortiserede omkostning for `Decrement` i operation i kan begrænses som:

$$\begin{aligned}
\hat{c}_i &= c_i + (\Phi(x_i) - \Phi(x_{i-1})) \\
&\leq k + k = 2k\ .
\end{aligned}$$

$\triangle$

d) Giv et eksempel på en sekvens af operationer som viser at det *ikke* er muligt at give en konstant (uafhængig af k) grænse på den amortiserede omkostning for både INCREMENT og DECREMENT.

Svar: Betragt sekvensen hvor der først laves 10^{k-1} INCREMENT operationer og derefter en lang skiftende sekvens af DECREMENT og INCREMENT operationer gentaget n gange. Det samlede antal ændringer for alle operationer er mindst $2nk$, da hver af de sidste $2n$ operationer kræver at alle decimaler ændres. Den samlede amortiserede omkostning for DECREMENT og INCREMENT er derfor mindst $2nk/(10^{k-1} + 2n)$, kommer vilkårligt tæt på k når $n \to \infty$. $\triangle$

2 Fibonacci hobe

Betragt en Fibonacci hob H (eng. *Fibonacci heap*), som beskrevet i CLRS digitalt kapitel 19, hvorpå der udføres n_1 Insert operationer, n_2 Extract-Min operationer, og n_3 Decrease-Key operationer. Det totale antal operationer er $n = n_1 + n_2 + n_3$. Hvilke af følgende udsagn er altid sande? Vælg ét eller flere korrekte svar og beskriv hvordan du kom frem til dem (både positive og negative svar).

1. Den samlede worst case tid for operationerne er $\Omega(n \lg n)$.
2. Den samlede worst case tid for operationerne er $O(n \lg n)$. ✓
3. Den samlede worst case tid for operationerne er $O(n_1 + n_3 + n_2 \lg n)$. ✓
4. H har $n_1 - n_2 - n_3$ knuder.
5. H har $m(H) = n_1 - n_2$ markerede knuder.

Svar: Den amortiserede analyse giver en øvre grænse på $O(n_1 + n_3 + n_2 \lg n)$ for operationerne. Da $n = n_1 + n_2 + n_3$ er det specielt $O(n \lg n)$. Hvis $n_1 + n_3 > n_2 \lg n$ bliver tiden $O(n)$, så tiden er ikke nødvendigvis $\Omega(n \lg n)$. Antal knuder i H er antal indsættelser minus antal sletninger, dvs. $n_1 - n_2$ som er forskellig fra $n_1 - n_2 - n_3$ hvis $n_3 > 0$. Antal markerede knuder kan ikke udtrykkes ved hjælp af n_1 og n_2 — for eksempel ændrer Insert ikke antallet af markerede knuder. △

3 Korrekthedsargumenter

Betragt følgende problem, kaldet `MaxProduct`:

Givet en liste af tal $x_1, \ldots, x_n \in \mathbf{R}$ (kan være både positive og negative), find en delmængde $I^* \subseteq \{1, \ldots, n\}$ hvor produktet $\prod_{i \in I^*} x_i$ er så stort som muligt, dvs. for alle mængder $I \subseteq \{1, \ldots, n\}$ skal gælde at $\prod_{i \in I} x_i \leq \prod_{i \in I^*} x_i$.

Eksempel. På input $x_1 = -4$, $x_2 = -0.5$, $x_3 = 0.5$, $x_4 = 0.5$, $x_5 = 1.5$, $x_6 = 6$ kan vi vælge $I = \{5, 6\}$ og få et produkt på $1.5 \cdot 6 = 9$, men et bedre valg er $I = \{1, 2, 5, 6\}$ som giver produktet $(-4) \cdot (-0.5) \cdot 1.5 \cdot 6 = 18$. Den tomme mængde $I = \emptyset$ giver per definition $\prod_{i \in \emptyset} x_i = 1$.

Hvad gælder altid for en optimal løsning I^*? Vælg ét eller flere korrekte svar og beskriv hvordan du kom frem til dem (både positive og negative svar).

1. I^* indeholder *ikke* noget indeks i hvor x_i er mellem 0 og 1 ($x_i \in (0, 1)$). ✓
2. I^* indeholder *alle* indekser i hvor $x_i < 0$.
3. I^* indeholder *alle* indekser i hvor $x_i > 1$. ✓
4. I^* indeholder *alle* indekser i hvor $|x_i| > 1$ (i.e., absolutværdien af x_i er større end 1).

Svar: Den optimale løsning har altid produkt mindst 1 fordi vi kan vælge $I = \emptyset$. Hvis I^* indeholdt i hvor $x_i \in (0,1)$ så kunne man opnå et større produkt ved at vælge $I = I^* \setminus \{i\}$, hvilket er en modstrid. Hvis antallet af negative indeks er ulige kan vi ikke have dem alle med i en optimal løsning, da produktet så ville blive negativt. Hvis vi udelader et indeks med $x_i > 1$, så $i \notin I^*$ kan vi opnå et større produkt ved at vælge $I = I^* \cup \{i\}$, hvilket er en modstrid. Hvis antallet af indeks i med $x_i < -1$ er ulige kan vi ikke have dem alle med i en optimal løsning, da produktet så ville blive negativt. $\triangle$

4 Induktionsbeviser

I denne opgave antager vi at T er en sorteret tabel med $n > 1$ elementer således at for $i < j$ gælder $T[i] \leq T[j]$. Antag at $1 \leq p \leq r \leq n$, hvor p, r, n er heltal. Betragt følgende funktion, i pseudo-kode notationen fra CLRS:

```
WhileMystery(x, T, p, r)
1  low = p
2  high = max(p, r + 1)
3  while low < high
4       mid = ⌊(low + high)/2⌋
5       if x ≤ T[mid]
6            high = mid
7       else
8            low = mid + 1
9  return high
```

a) Lad $high_i$ og low_i betegne værdien af variablene $high$ og low efter i iterationer af **while** løkken. Bevis ved induktion at $high_i - low_i \leq n/2^i$. (Du skal bruge antagelsen $1 \leq p \leq r \leq n$.) Der lægges vægt på, at argumentationen er klar og koncis.

Svar: Inden den første iteration har vi $high_0 - low_0 = \max(p, r+1) - p \leq n = n/2^0$. Det viser basistilfældet $i = 0$. I induktionsskridtet antager vi, at efter $i-1$ iterationer gælder $high_{i-1} - low_{i-1} \leq n/2^{i-1}$. Efter iteration i er vi i ét at to tilfælde afhængigt af udfaldes af sammenligningen i linje 5: $high_i = \lfloor (low_{i-1} + high_{i-1})/2 \rfloor$ og $low_i = low_{i-1}$ eller $low_i = \lfloor (low_{i-1} + high_{i-1})/2 \rfloor + 1$ og $high_i = high_{i-1}$. I det første tilfælde har vi, via induktionshypotesen, og fordi nedrunding kun gør forskellen mindre:

$$high_i - low_i = \lfloor (low_{i-1} + high_{i-1})/2 \rfloor - low_{i-1} \leq (high_{i-1} - low_{i-1})/2 \leq (n/2^{i-1})/2 = n/2^i \ .$$

I det andet tilfælde har vi, fordi $\lfloor x \rfloor + 1 \geq x$ for alle x:

$$high_i - low_i = high_{i-1} - (\lfloor (low_{i-1} + high_{i-1})/2 \rfloor + 1) \leq (high_{i-1} - low_{i-1})/2 \leq (n/2^{i-1})/2 = n/2^i \ .$$

$\triangle$