Algoritmer og datastrukturer - Skriftlig prøve 4-timer

330

11 april 2023

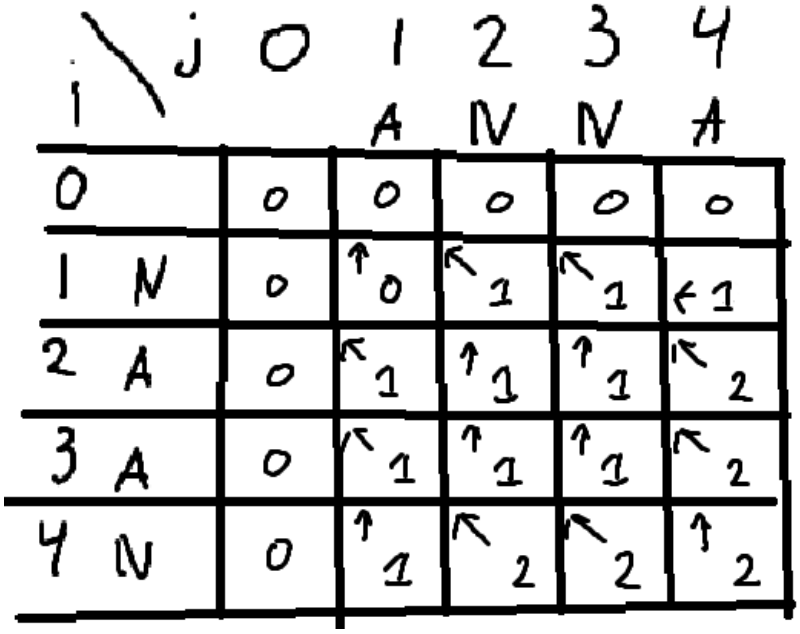
Planlagt: 12:00 - 16:00

Eksamensnr: 330

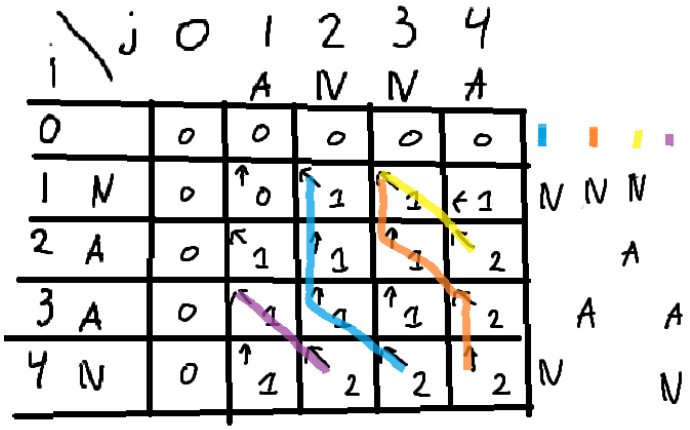
Plads: EH-0105

Opgave 19

Tabellen dannet ud fra algoritmen



Og med løsninger angivet:



Her ses det, at den længste delsekvens har længden 2, og kan laves på 3 måder: NN, NA og AN. Pilene angiver henholdsvis hvor deres sekvens kommer fra. En vandret eller lodret pil angiver at a_i!=a_j, og tager derfor maks af ”cellen” oppefra eller venstre (ved dem hvor de er lige store, tag blot pil op). Hvis derimod a_i==a_j, så tag cellen oppe til venstre, og læg en til (skrå pil), da dette er en tilføjelse til den delsekvens.

Opgave 20

a)

Benytter samme struktur som ved almindelig LCS, men i stedet for længden er jeg interesseret i maksimale sum af deres vægte. Derfor vælger jeg i case 2 i stedet for blot at lægge 1 til, så lægger jeg deres vægt til, og får derfor rekursionsligningen:

$$C[i,j] = \begin{cases} 0 & \text{if } i=0 \vee j=0 \\ C[i-1,j-1] + w[x_i] & \text{if } x_i = y_i \\ \max(C[i-1,j], C[i,j-1]) & \text{if } x_i \neq y_i \end{cases}$$

b)

Da intet af rekursionsligningen der har betydning for køretiden er ændret (da opslag i vægtfunktionen er givet i konstant tid), kan samme analyse fra LCS algoritmen benyttes og er derfor givet til $\Theta(n \cdot m)$ med brug af bottom up dynamisk programmering. Da vi ved, at vi maks skal finde n*m løsninger til C arrayet, og derfor n*m konstante operationer (ud fra rekursionsligningen) hvilket giver $\Theta(n \cdot m)$.

C)

Algoritmen kan nemt ændres, da der blot skal kontrolleres for, om den nuværende delsekvens bliver større ved tilføjelse af et nyt element. Dvs. i case 2, skal der blot tages max af c[i-1, j-i] og af c[i-1, j-1] + vægten. Den modificerede ligning vil blot blive:

$$C[i,j] = \begin{cases} 0 & \text{if } i=0 \vee j=0 \\ \max(C[i-1,j-1], C[i-1,j-1] + w[x_i]) & \text{if } x_i = y_i \\ \max(C[i-1,j], C[i,j-1]) & \text{if } x_i \neq y_i \end{cases}$$

Dette vil virke, da vi derfor blot vil ”undgå” de bogstaver/elementer der er negative. Fx hvis ABC er en given delsekvens, hvis B’s vægt da bliver negativ kan den blot fjernes fra delsekvensen og derfra vil AC stadig være en gyldig delsekvens, og derfor kan de elementer med negativ vægt blot ignoreres.

Opgave 21

Basecase for induktionsbeviset:

Basecase:

$high_0 - low_0 \leq \frac{n}{2^0} = n$

$high_0 \leq n$

$low \leq n$

og da $high_0 - \overbrace{low_0}^{1 \leq} \leq high_0$

må $high_0 - low_0 \leq n$

Herefter kan et induktionstrin laves (bemærk floor af noget altid er mindre eller lig)

Induktionstrin: i+1

$high_{i+1} - low_{i+1} \leq \frac{n}{2^{i+1}}$

ved hver iteration må

enten

$high_{i+1} = \frac{low_i + high_i}{2}$ og $low_{i+1} = low_i$

eller

$low_{i+1} = \frac{low_i + high_i}{2} + 1$ og $high_{i+1} = low_i$

Vi kan derfor lave 2 cases, et hvor high_{i+1} ændres og en hvor low_{i+1} gør:

$high_{i+1} = \frac{low_i + high_i}{2}$, $low_{i+1} = low_i$

$\frac{low_i + high_i}{2} - low_i \leq \frac{n}{2^{i+1}}$

$low_i + high_i - 2low_i \leq \frac{n}{2^i}$

$high_i - low_i \leq \frac{n}{2^i}$ (induktions hypotese)

$$low_{i+1} = \frac{low_i + high_i}{2} + 1 \quad , \quad high_{i+1} = high_i$$

$$high_i - \left(\frac{low_i + high_i}{2} + 1\right) \leq \frac{n}{2^{i+1}}$$

$$2high_i - low_i - high_i - 1 \leq \frac{n}{2^i}$$

$$high_i - low_i - 1 \leq \frac{n}{2^i} \quad \text{(induktions hypotese)}$$

Da det nu er vist for begge cases, er det derfor bevist ved induktion.

Opgave 22 er på næste side (s. 8) grundet pladsmangel.

Opgave 22

a)

Hvis I_* kun indeholder negative elementer, da vil der altid skulle være skulle tilføjes negative tal i par, da -x*-y=y*x. Hvis der tilføjes et uligt antal negative tal, vil det endelige produkt være negativt, og altså derfor mindre end 0, og da vil den optimale løsning være den tomme mængde, da den per definition gav 1.

b)

Da elementerne er sorteret, kan vi blot grådigt tilføje dem til vores sæt således:

```
I:=[]
i:=1
While i<n:
  if X[i]>1:
     add i to I
     i+=1
  else if X[i]<0 and X[i+1]<0 and X[i]*X[i+1]>1:
     add i and i+1 to I
     i+=2
return I
```

Dvs. vi ser hver gang om x_i er over 1, hvis da, tilføj blot til I. Da et produkt af noget med et tal over 1 altid vil blive større.
Hvis x_i derimod er negativ, og den næste er negativ samt at deres produkt er over 1, så tilføj da begge til I. Dette gøres, da der altid skal være et lige antal negative tal i I for at få et positivt produkt, og da tallene er sorteret er vi sikre på at de største ”produktpar” af de negative værdier (og her lægges 2 til i, da vi allerede har tilføjet i+1).
Dvs. vi gennemgår maksimalt alle n tal, og får derfor en tidskompleksitet på O(n).

AD eksamen 2023-04-11

Algoritmer og datastrukturer - Skriftlig prøve 4-timer (NDAA04011E) 11-04-2023 12:00

330

1 MergeSort og rekursion

Hvis MergeSort(A,p,r), implementeret som i CLRS sektion 2.3, [...]

5 7 samtidige kald

2 Korteste-veje træ

Betragt et korteste veje problem (eng. single-source shortest-paths problem) på en orienteret graf G=(V,E) med vægtfunktion w: E->R og startknude s. [...]

1 Enhver sti i G' er en korteste vej i G.

3 Korteste veje

Betragt korteste veje problemet (eng. single-source shortest-paths problem) med startknude a på følgende graf: [...]

10 (e,f)

3 (a,d)

2 (a,c)

5 (c,b)

7 (c,e)

4 Prim's algoritme

Betragt følgende vægtede, uorienterede graf: [...]

1 a, b, d, e, f, c

5 Mindste udspændende træ

Betragt mindste udspændende træ (eng. minimum spanning tree) problemet på følgende vægtede, uorienterede graf: [...]

9 {d,f}

10 {e,f}

6 {c,d}

1 {a,b}

4 {b,d}

6 Snit og sikre kanter

Betragt følgende uorienterede, vægtede graf med knudemængde V={a,b,c,d,e,f}:
[...]

4 {d,f} er en sikker kant for A.

5 {c,e} er en sikker kant for A.

3 Lad S_3={b,e,f}. Snittet (S_3,V - S_3) respekterer kantmængden A.

7 Amortiseret analyse

I denne opgave betragter vi potentialfunktionen der bruges til at analysere en binær tæller i CLRS sektion 17.3. [...]

1 Phi(D_i) = Omega(lg i).

4 Phi(D_i) = O(lg i).

5 Phi(D_i) = O(i).

8 Køretid

Betragt følgende kode i pseudo-kode notationen fra CLRS: [...]

5 T(n) = Omega(n lg n)

6 T(n) = Omega(n^2)

3 T(n) = O(n^2)

4 T(n) = Omega(n)

9 Rekursionsligninger

Hvilke af disse rekursionsligninger har løsningen [...]

5 T(n) = 3 T(floor(n/3)) + n.

6 T(n) = T(n-1) + lg n.

2 T(n) = 2 T(floor(n/2)) + 3n.

10 Del og hersk

Antag at du har en rekursiv algoritme A, og lad n betegne størrelsen på algoritmens input X. [...]

3 T(n) = 4 T(floor(n/2)) + O(n^2).

11 Fibonacci hobe

Betragt en Fibonacci hob H (eng. Fibonacci heap), som beskrevet i CLRS kapitel 19, hvorpå der udføres n_1 Insert operationer, n_2 Extract-Min operationer, og n_3 Decrease-Key operationer. [...]

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).

12 Binære søgetræer

Betragt et rød-sort binært søgetræ T (eng. red-black binary search tree), som beskrevet i CLRS kapitel 13. [...]

5 En sort knude med nøglen 30.

3 En rød knude med nøglen 58.

13 Binære søgetræer

Betragt dette rød-sorte binære søgetræ (igen med bladene udeladt): [...]

2 42 indsættes som en rød knude under knuden 47.

14 Disjunkte mængder

Vi betragter trærepræsentationer af disjunkte mængder (eng. disjoint sets), som beskrevet i CLRS sektion 21.3, ved brug af heuristikkerne union by rank og path compression. [...]

2 k kald til Find-Set(42) lige efter hinanden tager tid O(k + lg n).

3 Kaldene Make-Set(x_1),…,Make-Set(x_k) tager total tid O(k).

15 Invarianter

Betragt følgende funktion, i pseudo-kode notationen fra CLRS, hvor input A er en tabel af heltal og n er længden af A. [...]

3 r >= m.

5 Mindst 1 element i A[1],…,A[i-1] er mindre end eller lig med r.

1 m = min(A[1],…,A[i-1]).

16 Valg af datastruktur

Antag at vi gerne vil skabe en datastruktur Ventetid, der indeholder en mængde af n afgangstidspunkter (ugedag og klokkeslet, fx for en bus). [...]

4 Fibonacci hob (eng. Fibonacci heap), tid O(1).

17 Korrekthedsargumenter

Betragt følgende problem, kaldet MaxProduct: [...]

3 I* indeholder alle indekser i hvor x_i > 1.

1 I* indeholder ikke noget indeks i hvor x_i er mellem 0 og 1 (x_i tilhører (0,1)).

18 Dynamisk programmering

En stigende delsekvens af en tabel A, der indeholder heltal A[1],…,A[n], [...]

4 Længden af en længste ikke-aftagende delsekvens.