Prøveeksamen i Algoritmer og Datastrukturer (NDAA04010U)
(version med svar og forklaringer)

Københavns Universitet

2023-03-27

Instruktioner

Du skal besvare 13 opgaver, beskrevet nedenfor. Karakteren baseres på det samlede antal point (vægten af hver opgave er angivet) og på en helhedsbedømmelse. **Sættet svarer i omfang til ca. 2/3 af en 4-timers eksamen, dvs. forventes at kunne besvares på 160 minutter. Der er blot afsat 60 minutter til prøveeksamen i ITX lokalet, så du bør forberede dine svar inden. Dine besvarelser bliver *ikke* gjort synlige for underviserne og altså heller ikke rettet. Vejledende besvarelse findes på Absalon.** Opgavesættet har 10 nummererede sider. CLRS refererer til kursets lærebog, Introduction to Algorithms, 3rd edition.

Skriftlige hjælpemidler samt brug af lokalt installeret software er tilladt. Det er *ikke* tilladt at tilgå internet eller at kommunikere med andre.

Multiple-choice opgaver besvares i ITX systemets multiple-choice modul. I hvert multiple-choice spørgsmål er det angivet hvorvidt der er præcis én korrekt svarmulighed eller der skal angives en *mængde* af ét eller flere korrekte svar. Ved retning af multiple-choice opgaver gives der negative point for forkerte svar. Eksempler:

- Opgave X har 8 svarmuligheder, hvoraf 4 er korrekte. Hvert kryds ved en korrekt svarmulighed giver 1 point, og hvert kryds ved en forkert svarmulighed giver -1 point. Hvis du fx sætter kryds ved alle de korrekte svarmuligheder og én forkert svarmulighed får du 3 point. Hvis du ikke sætter nogen krydser får du 0 point, og sætter du krydser ved alle svarmuligheder får du også 0 point.

- Opgave Y har 5 svarmuligheder, hvoraf 1 er korrekt. Kryds ved den korrekte svarmulighed giver 4 point, og kryds ved en forkert svarmulighed giver -1 point.

Svar på skriftlige opgaver afleveres som et Word dokument i ITX systemet. Der er mulighed for at tegne og skrive på tablet, og indsætte i dokumentet. Efter forsiden skal der laves *én side til hvert delspørgmål, i samme rækkefølge som i opgavesættet.* Angiv spørgsmålets nummer og bogstav (fx 10.a) men kopiér ikke opgaveteksten ind i besvarelsen. Totalt skal Word dokumentet have 8 sider. (Hvis du ikke svarer på et spørgsmål, så lav en tom side.)

Multiple-choice

Dine svar skal indtastes i ITX-systemets multiple-choice modul.

1 MergeSort og sammenligninger (4%)

Betragt følgende input til MERGESORT, implementeret som i CLRS sektion 2.3.

42	33	36	43	11	38	22	66

Hvor mange gange sammenligner MERGESORT elementet 42 med et andet element i input? (Sammenligninger med ∞ skal ikke tælles med.) Vælg præcis ét svar.

1. 3 sammenligninger
2. 4 sammenligninger
3. 5 sammenligninger ✓
4. 6 sammenligninger
5. 7 sammenligninger

Svar: Vi kan fokusere på de sorterede lister hvor 42 indgår. Der laves 1 sammenligning for at lave den sorterede liste med 2 elementer, 2 sammenligninger for at lave den sorterede liste med 4 elementer, og yderligere 2 sammenligninger for at lave den endelige sorterede liste. Totalt 5 sammenligninger.

2 MergeSort (4%)

Antag at MERGESORT(A, p, r), implementeret som i CLRS sektion 2.3, bliver kaldt på 21 elementer (dvs. $r - p + 1 = 21$). Hvor mange kald bliver der totalt lavet til MERGE-SORT? Vælg præcis ét svar.

1. 11
2. 21
3. 31
4. 41 ✓
5. 51

Svar: Antal elementer i hvert kald af mergesort, opdelt efter rekursionsniveau, er: 21; 10, 11;5, 5, 5, 6; 2, 3, 2, 3, 2, 3, 3, 3; 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 2; 1, 1, 1, 1, 1, 1, 1, 1, 1, 1. Antal kald er altså $1 + 2 + 4 + 8 + 16 + 10 = 41$. Alternativt kan man se at antal indre knuder i et binært træ med 21 blade altid er $2 \cdot 21 - 1 = 41$.

3 Korteste-veje træ (4%)

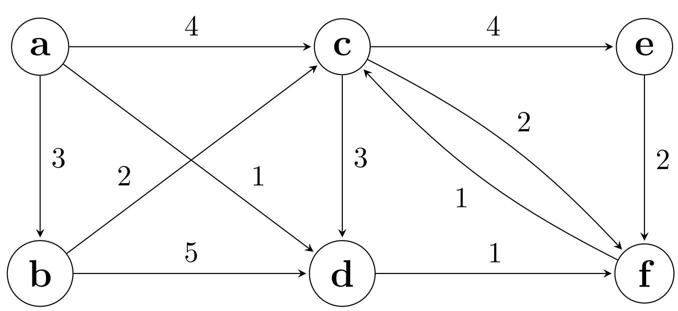
Betragt et korteste veje problem (eng. *single-source shortest-paths problem*) på en graf $G = (V, E)$ med vægtfunktion $w : E \to \mathbb{R}$ og startknude s. Hvilke af følgende egenskaber har et korteste-veje træ (eng. *shortest-paths tree*) $G' = (V', E')$ altid? Vælg ét eller flere korrekte svar.

1. I G' kan alle knuder i V nås fra s.
2. Knuden s har kun udgående kanter i G'. ✓
3. G' har $|V'| - 1$ kanter. ✓
4. Hvis alle vægte er unikke indeholder G' af de $|E'|$ kanter i G, der har lavest vægt.
5. G' indeholder den kant i G, der har lavest vægt.

Svar: G' forbinder kun s med de knuder knuder i V hvortil der findes en vej i G. Der er kun udgående kanter fra s fordi træet ikke har nogen cykler. G' er et træ, og derfor er antal kanter lig med antal knuder minus 1. En lav vægt er ikke en garanti for at være med i en korteste vej fra s — for eksempel kan visse kanter ikke nås fra s.

4 Korteste veje (4%)

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



Hvilke kanter er med i korteste-vej træet? Vælg ét eller flere korrekte svar.

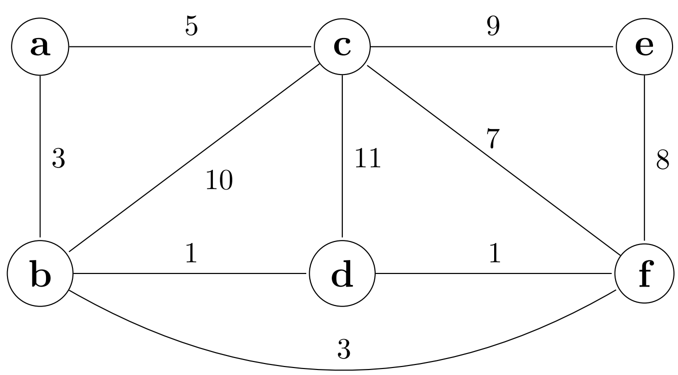
1. (a,b) ✓
2. (a,c)
3. (a,d) ✓
4. (b,c)
5. (b,d)

6. (c,d)
7. (c,e) ✓
8. (c,f)
9. (d,f) ✓
10. (e,f)
11. (f,c) ✓

Svar: Da der kun er positive vægte kan kanterne i træet findes ved at køre Dijkstra's algoritme, der giver det angivne output (som i dette tilfælde er unikt).

5 Mindste udspændende træ (4%)

Betragt mindste udspændende træ (eng. *minimum spanning tree*) problemet på denne graf:



Hvilke kanter er med i mindste udspændende træ? Vælg ét eller flere korrekte svar.

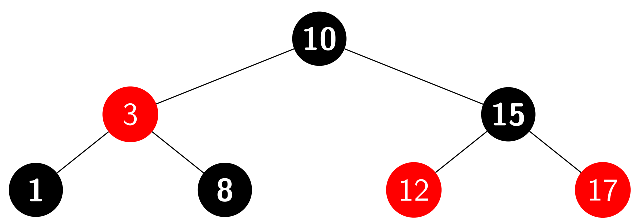
1. {a,b} ✓
2. {a,c} ✓
3. {b,c}
4. {b,d} ✓
5. {b,f}
6. {c,d}
7. {c,e}
8. {c,f}

9. {d,f} ✓
10. {e,f} ✓

Svar: Mindste udspændende træ kan findes ved at køre fx Kruskál's eller Prim's algoritme og giver det angivne input (som i dette tilfælde er unikt).

6 Rød-sorte søgetræer (4%)

Betragt dette rød-sorte binære søgetræ (eng. *red-black binary search tree*), hvor bladene er udeladt på tegningen:



Antag at vi bruger indsættelsesalgoritmen RB-INSERT fra CLRS kapitel 13 til at indsætte nøglen 9. Hvad sker med træet? Vælg præcis ét svar.

1. 9 indsættes som højre barn til knude 8 og farves rød. ✓
2. 9 indsættes som højre barn til knude 8 og farves sort.
3. 9 indsættes som venstre barn til knude 12 og farves rød.
4. 9 indsættes som venstre barn til knude 12 og farves sort.
5. Ingen af ovenstående.

Svar: RB-INSERT finder det rette sted i træet til nøglen 9, som det højre barn under knude 8. Knuden farves rød og derefter kaldes RB-INSERT-FIXUP, men den terminerer uden at ændre noget fordi forældreknuden 8 er sort.

7 Køretid for rød-sorte søgetræer (4%)

Betragt et rød-sort binært søgetræ T (eng. *red-black binary search tree*), som beskrevet i CLRS kapitel 13, hvorpå der udføres n_1 INSERT operationer, n_2 DELETE operationer, og n_3 SEARCH operationer (sidstnævnte kaldes også "access" 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.

1. Den samlede tid for operationerne er $\Omega(N)$. ✓

2. Den samlede tid for operationerne er $O(N \lg N)$. ✓

3. Den samlede tid for operationerne er $O(n_3 + (n_1 + n_2) \lg N)$.

4. Dybden af T er højst $2 \lg(n_1 + 1)$. ✓

5. Alle blade i T er i dybde mindst $\lfloor \lg(n_3 + 1) \rfloor$.

6. Alle sorte knuder i T har en afstand til roden, der er et lige tal.

Svar: Hver operation tager mindst konstant tid, så den samlede tid er $\Omega(n)$. Antal knuder n i træet er på ethvert tidspunkt højst $n_1 \leq N$. Hver operation tager altså højst $O(\lg N)$ tid, så den samlede tid er $O(n \lg N)$. I værste fald tager søgeoperationerne tid $\Omega(\lg n_1)$, dvs. tid $\Omega(n_3 \lg n_1)$ totalt, og tiden kan altså ikke begrænses af $O(n_3 + (n_1 + n_2) \lg N)$. Et rød-sort binært søgetræ kan have blade i dybde 2, bladenes dybde er ikke større end $\lfloor \lg(n_3 + 1) \rfloor$ generelt. Sorte knuder kan optræde på alle niveauer.

8 Amortiseret analyse (4%)

I denne opgave betragter vi potentialfunktionen der bruges til at analysere dynamiske tabeller i CLRS sektion 17.4.1. Lad Φ_i betegne potentialfunktionens værdi efter i TABLE-INSERT operationer, uden nogen TABLE-DELETE operationer. Hvilke egenskaber har Φ_i? Vælg ét eller flere korrekte svar.

1. $\Phi_i = \Omega(1)$.✓

2. $\Phi_i = O(1)$.

3. $\Phi_i = \Omega(i)$.

4. $\Phi_i = O(i)$.✓

5. $\Phi_i \geq \Phi_{i-1}$.

6. $\Phi_i \geq 0$. ✓

Svar: $\Phi_i = 2 \cdot num_i - size_i$, hvor $num_i = i$ og $size_i \geq 1$ er den aktuelle kapacitet. Vi har altid $\Phi_i \leq i/2$, så $\Phi_i = O(i)$, men lige efter en tabeludvidelse har vi $\Phi_i \leq 2$, dvs. det gælder ikke at $\Phi_i = \Omega(i)$. Lige efter en tabeludvidelse har vi $\Phi_i < \Phi_{i-1}$. Potentialfunktionen er defineret så den er ikke-negativ, og den er positiv fra første skridt og fremefter.

9 Del og hersk (4%)

Hvilke af disse rekursionsligninger har løsningen $T(n) = \Theta(n^2)$? Antag at $T(n) = 1$ for $n \leq 1$. Vælg ét eller flere korrekte svar.

1. $T(n) = 4T(\lfloor n/2 \rfloor) + n \lg n$. ✓
2. $T(n) = 4T(\lfloor n/2 \rfloor) + n^2$.
3. $T(n) = 2T(\lfloor n/4 \rfloor) + n^2$. ✓
4. $T(n) = 9T(\lfloor n/3 \rfloor) + n^3$.
5. $T(n) = T(n-1) + n^2$.
6. $T(n) = T(n-1) + n$. ✓

Svar: De fire første rekursionsligninger kan løses ved hjælp af master theorem, hvor nummer 1 og 3 falder i henholdsvis tilfælde 1 og 3 som giver $T(n) = \Theta(n^2)$. De sidste ligninger kan løses med substitutionsmetoden. I ligning 5 er der n led hvor halvdelen har størrelse mindst $(n/2)^2 = n^4/4$, så $T(n) = \Omega(n^3)$. I ligning 5 er der n led af størrelse højst n, hvor halvdelen har størrelse mindst $n/2$, så $T(n) = \Theta(n^2)$.

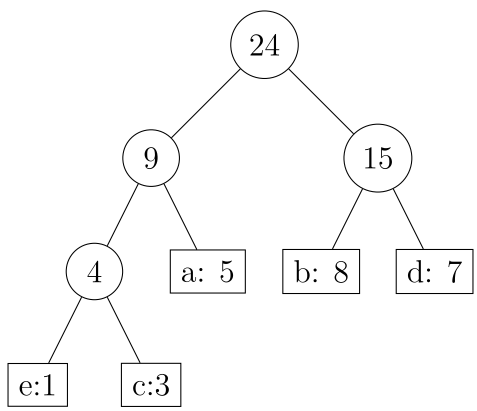
Skriftlige opgaver

10 Huffman tree (4%)

Vi betragter algoritmen til konstruktion af Huffman koder i CLRS sektion 16.3.

a) Tegn Huffman træet som algoritmen beregner for alfabetet $C = \{a, b, c, d, e\}$ med frekvenser $f_a = 5$, $f_b = 8$, $f_c = 3$, $f_d = 7$, $f_e = 1$. Det er ikke nødvendigt at angive labels (0 og 1) på kanterne, men angiv frekvenser i blade såvel som indre knuder (se eksempel i CLRS figur 16.4). Skriv dit svar på side 2 i Word dokumentet.

Svar: Algoritmen beregner dette træ (tegnet uden 0-1 labels på kanterne):



11 Dynamisk programmering (10%)

Vi betragter følgende variant af *rod cutting* problemet fra CLRS sektion 15.3. Input er et heltal n og en vektor p hvor p_i er fortjenesten på en stav af længde n. I lighed med rod cutting skal en stav (eng. *rod*) af længde n deles op i kortere stave, alle af heltalslængde, således at den totale fortjeneste maksimeres. I modsætning til det originale rod cutting problem er der dog en omkostning på 1 for hvert snit, som skal modregnes i fortjenesten.

Eksempel. Antag at vi har en stav af længde $n = 5$, hvor $p = (p_1, p_2, p_3, p_4, p_5) = (1, 3, 5, 6, 6)$. Hvis vi deler staven i to dele af størrelse henholdsvis 2 og 3, så bliver fortjenesten $p_2 + p_3 - 1 = 7$, hvor vi trækker 1 fra fordi der er ét snit. Hvis vi i stedet delte i 5 dele af størrelse 1 ville fortjenesten blive $5\,p_1 - 4 = 1$.

a) Skriv en rekursionsligning for fortjenesten (eng. *revenue*) r_n for en stav af længde n. Skriv dit svar på side 3 i Word dokumentet.

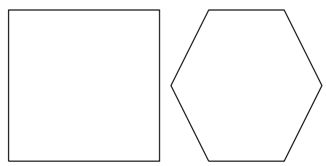
Svar: Vi har $r_0 = 0$ og med samme argument for standard rod cutting er $r_n = \max_{0 \le i \le n}(p_i - r_{n-i} - 1)$, hvor vi trækker 1 fra pga. omkostningen ved snittet.

b) Lav en analyse af køretiden af algoritme, der bruger rekursionsligningen og dynamisk programmering til at beregne fortjenesten ved en stav af længde n. Analysen skal give en øvre grænse på køretiden som funktion af n i store-O notation, der er så præcis som mulig. Skriv dit svar på side 4 i Word dokumentet.

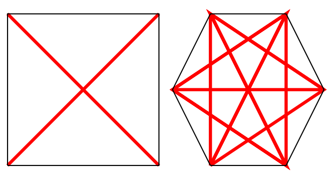
Svar: Analysen for rod cutting kan genbruges, da rekursionsligningen har samme struktur. Ændringen påvirker kun værdierne og tiden for hvert skridt øges med højst en konstant faktor. Derfor er køretiden $O(n^2)$.

12 Induktionsbeviser (6%)

I denne opgave betragter vi n-kanter som disse eksempler ($n = 4$ og $n = 6$):



Vi er interesserede i antal *diagonaler*, dvs. linjer der forbinder to hjørner i n-kanterne og er helt indeholdt i figuren. En 3-kant har ingen diagonaler da alle hjørner er naboer, mens vi for $n = 4$ og $n = 6$ har henholdsvis 2 og 9 diagonaler.



En diagonal deler en n-kant en n_1-kant og en n_2 kant, der deler en kant og hvor $n_1 + n_2 = n + 2$. For eksempel deler 4-kantens diagonal den i to 3-kanter.

a) Bevis ved induktion at antal diagonaler d_n i en n-kant er lig med $n(n-3)/2$ for $n \geq 3$. Der lægges vægt på, at argumentationen er klar og koncis. Skriv dit svar på side 5 i Word dokumentet.

Svar: For $n = 3$ har vi $d_n = 0 = 3(3-3)/2$, hvilket etablerer basistilfældet. Antag nu at $n > 3$, at udsagnet er korrekt for n'-kanter med $n' < n$, og betragt en n-kant. Tag tre hjørner v_1, v_2, v_3 der ligger ved siden af hinanden, dvs. hvor v_1 og v_3 er forbundet til v_2. Fra v_2 kan der laves diagonaler til hver af de andre $n-3$ hjørner og desuden kan der laves en diagonal mellem v_1 og v_3. De resterende diagonaler ligger i den $(n-1)$-kant, der fås ved at udelade v_2 (og forbinde v_1 og v_3 direkte). Ifølge induktionshypotesen har vi $d_{n-1} = (n-1)(n-4)/2$ diagonaler i denne del, dvs.

$$d_n = n - 3 + 1 + d_{n-1} = n - 2 + (n-1)(n-4)/2 = n(n-3)/2 \ .$$

13 Korrekthed og køretid (10%)

I denne opgave bruger vi matrix og vektor-notation, se evt. CLRS appendix D.1. Fibonaccitallene $F_0, F_1, F_2, \ldots$ er defineret ved $F_0 = 0$, $F_1 = 1$, og $F_n = F_{n-1} + F_{n-2}$ for $n > 1$. Betragt nu matricen A og søjlevektorer på formen $x = (x_1, x_2)^T$.

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \qquad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

a) Argumentér for at for $x = (F_{n-2}, F_{n-1})^T$ opfylder matrix-vektor produktet ligheden $Ax = (F_{n-1}, F_n)^T$ for $n > 1$. Skriv dit svar på side 6 i Word dokumentet.

Svar: Per definition af matrix-vektor produkter har vi:

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_{n-2} \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} F_{n-1} \\ F_{n-2} + F_{n-1} \end{pmatrix} = \begin{pmatrix} F_{n-1} \\ F_n \end{pmatrix},$$

hvor den sidste lighed bruger definitionen af F_n.

b) Bevis at for $x = (0, 1)^T$ gælder ligheden $A^{n-1}x = (F_{n-1}, F_n)^T$ for $n \geq 1$. Skriv dit svar på side 7 i Word dokumentet.

Svar: Bevis ved induktion i n. Basistilfældet $n = 1$ gælder da $A^{n-1}x = x = (F_0, F_1)^T$. Antag som induktionshypotese at $A^{i-1}x = (F_{i-1}, F_i)^T$ for $1 \leq i < n$. Da har vi

$$A^{n-1}x = A(A^{n-1}x) = A(F_{i-1}, F_i)^T = (F_{n-1}, F_n)^T,$$

hvor lighed nummer 2 bruger induktionshypotesen og lighed nummer 3 bruger del a).

c) Argumentér for at A^{n-1}, og dermed også $A^{n-1}x$, kan udregnes i tid $O(\lg n)$, hvis vi antager at aritmetiske operationer tager konstant tid. Skriv dit svar på side 8 i Word dokumentet.

Svar: Vi har følgende rekursionsligning:

$$A^n = \begin{cases} I & \text{hvis } n = 0 \\ (A^{n/2})^2 & \text{hvis } n > 0 \text{ er lige} \\ A(A^{(n-1)/2})^2 & \text{hvis } n \text{ er ulige} \end{cases}$$

Da A^n er af konstant størrelse betyder det at A^n kan udregnes i konstant tid ud fra enten $A^{n/2}$ eller $A^{(n-1)/2}$. Det giver rekursionsdybde højst $\lg(n)$, og dermed tid $O(\lg n)$.