Algoritmer og Datastrukturer

Datalogisk Institut, Københavns Universitet,
Pawel Winter & Christian Wulff-Nilsen

24-timers hjemmeprøve, Digital Eksamen, 1. april kl. 9:00 til 2. april kl. 9:00, 2020

Alle hjælpemidler så som kursusbogen og noter må benyttes under eksamen. Alle elektroniske hjælpemidler kan benyttes. Besvarelsen kan være enten på dansk eller engelsk og skal helst skrives i LaTeX, OpenOffice, Word eller lignende. Dog tillades også håndskrevne besvarelser, såfremt de er letlæselige for censor og eksaminator. Figurer må gerne laves i hånden. Det er valgfrit, om man vil digitalisere eventuelle håndskrevne dele af besvarelsen via scan eller via fotografi med kamera, smartphone eller lignende. **Aflevering er i Digital Eksamen.** Angiv eksamensnummer men ikke navn på første side i afleveringen. **Krav til afleveringsformat:**

- **en enkelt pdf indeholdende din besvarelse,**
- **ingen side i pdf'en må indeholde dele af svar fra mere end en delopgave; start derfor med en ny side for hver delopgave (og ikke kun for hver opgave). Besvarelsen af en delopgave skal være på hinanden følgende sider i pdf'en (eks.: side 3–5 er fint for samme delopgave, men ikke side 3, 5 og 6).**

Bemærk: deadline for aflevering er 2. april kl. 9:00 om morgenen. Forsinkede afleveringer accepteres ikke. Sørg for at aflevere i god tid. Der er mulighed for at uploade afleveringer flere gange indtil deadline. Skulle der mod forventning være problemer med at aflevere i Digital Eksamen tæt på deadline, send da straks en e-mail til `uddannelse@diku.dk` og `koolooz@di.ku.dk` med en beskrivelse af, hvad der er gået galt, og sørg for som vedhæftninger til denne e-mail at have besvarelsen som pdf samt et screendump, der viser problemet med at uploade i Digital Eksamen. Din besvarelse vil blive vurderet, hvis du har en gyldig begrundelse for ikke at aflevere i Digital Eksamen, og hvis du afleverer inden deadline.

Det er ikke tilladt at kommunikere med andre omkring sættet under eksamen. Den eneste undtagelse er spørgsmål omkring uklarheder i formuleringen af eksamenssættet, som kan stilles på Absalon i tråden "Spørgsmål til eksamenssættet" eller via e-mail til Christian (`koolooz@di.ku.dk`). Eventuelle rettelser eller kommentarer til eksamenssættet under eksamen vil blive givet via Announcements på Absalon.

Fuld besvarelse er besvarelse af alle opgaver. De enkelte opgavers og delopgavers vægt ved bedømmelsen er angivet i procent. Der må gerne refereres til algoritmer, datastrukturer og resultater fra pensumdelen af kursusbogen, dog ikke resultater i Exercises eller Problem-opgaverne i bogen. Specielt må man gerne begrunde en påstand med at henvise til, at det umiddelbart følger fra et bestemt lemma, en sætning eller lignende i pensumdelen af kursusbogen. **Det er ikke tilladt at basere sine svar på andre kilder end pensumdelen af kursusbogen.**

ADVARSEL: eksamen er individuel. Ved mistanke om eksamenssnyd har vi pligt til at melde dette til studielederen. Eksamenssnyd kan føre til bortvisning fra universitetet. Udsæt ikke dig selv eller dine medstuderende for denne risiko!

Dette eksamenssæt består af 4 opgaver og er på 14 sider, inkl. denne forside.

Version med vejledende løsninger

Opgave 1 (15%)

Denne opgave omhandler datastrukturer.

Del 1.1 (8%) Forklar overordnet, hvad en Fibonacci heap er, hvilke algoritmer inden for pensum denne datastruktur anvendes i. Kom herunder ind på køretider, både for udvalgte Fibonacci heap-operationer og for de algoritmer, der anvender Fibonacci heaps. Der forventes ikke nogen detaljeret gennemgang af algoritmerne eller operationerne. Brug gerne figurer og undlad beviser. Eventuelle håndkørsler skal være på små instanser. Teksten i din besvarelse skal fylde max en side; der er ingen grænse på, hvor meget figurer må fylde.

Løsning til 1.1 Her gives der f.eks. point for (i ikke-prioriteret rækkefølge):

1. at forklare i ord, hvad udvalgte Fibonacci heap-operationer gør,
2. at nævne køretider for udvalgte operationer, og at de er amortiserede,
3. at omtale anvendelser og køretider for Dijkstra/Prim (og prioritetskø for Huffman, selvom man lige så godt kan bruge binære min-heaps her),
4. at nævne, hvad der gør disse to algoritmer asymptotisk hurtigere med Fibonacci heaps end med binære heaps (DecreaseKey),
5. at forklare den overordnede struktur af en Fibonacci heap (træer med min-heap-egenskab, ordnet i rodliste), gerne med figur,
6. at nævne potentiale-metoden og potentiale-funktionen for Fibonacci heaps,
7. håndkørsel af udvalgte operationer.

I de resterende delopgaver fokuserer vi på amortiseret analyse.

I kapitlet "Amortized Analysis" i kursusbogen gennemgås eksemplet med en binær tæller. I denne opgave betragter vi i stedet en tæller for decimaltal. Tælleren består af k cifre hvert mellem 0 og 9 og disse repræsenteres ved et 0-indekseret array A med k indgange, som til at starte med alle er lig 0. Der udføres nu n INCREMENT-operationer på A, og hver operation øger decimaltallet i tælleren med 1. Hvis alle cifre i tælleren er 9, vil en INCREMENT-operation ændre disse cifre til 0.

Del 1.2 (2%) Giv pseudo-kode for INCREMENT og forklar kort i ord, hvordan denne procedure fungerer. Undlad et korrekthedsbevis.

Løsning til 1.2 Samme pseudo-kode som på side 454 i kursusbogen, bortset fra at der testes, om $A[i]$ er 9 i linje 2, og linje 6 ændres, så $A[i]$ øges med 1. Proceduren scanner tælleren fra det mindste til det mest betydende ciffer, indtil et ciffer findes med værdi mindre end 9. Alle scannede 9-taller sættes lig 0, og det fundne ciffer øges med 1. I tilfælde af overflow sættes alle cifre lig 0.

Point for:

1. korrekt pseudo-kode,
2. kort gennemgang af INCREMENT i ord

Del 1.3 (5%) Vis at worst-case-køretiden for INCREMENT er $O(k)$, og brug accounting-metoden til at vise, at den amortiserede køretid for INCREMENT er $O(1)$.

Løsning til 1.3 Worst-case-tiden er $O(k)$, idet INCREMENT bruger konstant tid pr. ciffer, og i værste fald scannes alle k cifre i A.

For at vise $O(1)$ amortiseret køretid betales en amortiseret omkostning på to credits, hver gang et ciffer øges. Den ene credit betaler for at øge cifferet, og den anden credit placeres på cifret (alternativt: hav amortiseret omkostning 1, hvis cifferet øges til en værdi under 9 og ellers en amortiseret omkostning på 2). Et ciffer, der opdateres fra 9 til 0, har amortiseret omkostning 0; den credit, der ligger på cifret, betaler for opdateringen. Vi kan nu analysere den samlede amortiserede omkostning af en INCREMENT-operation: højst et ciffer øges i løbet af operationen og alle andre ændringer i cifre har amortiseret omkostning 0. Derfor er den samlede amortiserede omkostning for en INCREMENT-operation højst 2.

Bemærk at antal credits aldrig kan være negativt, så den samlede $O(n)$ amortiserede omkostning for alle n operationer er en øvre grænse på den samlede faktiske omkostning.

Point for:

1. korrekt argument for worst-case-køretid: konstant tid pr. ciffer og k cifre ialt (evt. suppleret med worst-case-eksemplet $99\ldots9 \to 00\ldots0$),
2. korrekt amortiseret omkostning for ændring af et enkelt ciffer (2 eller 0),
3. forklaring af, hvor credits placeres (på de enkelte cifre, når de øges),
4. argument for at der højst kan være et ciffer, der øges i en INCREMENT-operation,
5. argument for, at antal credits ikke på noget tidspunkt kan være negativ, så den samlede amortiserede omkostning er mindst den samlede faktiske omkostning.

Opgave 2 (31%)

Denne opgave omhandler paradigmet Dynamisk Programmering.

Del 2.1 (8%) Giv en overordnet beskrivelse af dette paradigme og kom ind på det, du vurderer er det vigtigste inden for emnet Dynamisk Programmering. Brug gerne figurer og undlad beviser. Eventuelle håndkørsler skal være på små instanser. Teksten i din besvarelse skal fylde max en side; der er ingen grænse på, hvor meget figurer må fylde.

Løsning til 2.1 Point gives f.eks. for:

1. forklaring af hovedidéen i DP (gem løsninger til delproblemer og brug tabelopslag, hvis de skal løses igen),
2. beskrivelse (gerne overordnet) af optimal delstruktur, og hvorfor dette er en del af korrekthedsbeviset for en DP-algoritme,
3. beskrivelse af overlappende delproblemer,
4. forklaring af forskel mellem bottom-up og top-down med memoization (evt. fordele/ulemper ved de to),
5. eksempler på rekursionsligninger (rod-cutting, LCS),
6. optimal delstruktur skitseret for rod-cutting eller LCS,
7. rekursionsligninger for rod-cutting eller LCS,
8. håndkørsel af rod-cutting/LCS,
9. køretid og plads for rod-cutting/LCS,
10. hvordan man kommer fra løsningsværdi til løsning (f.eks. hvordan man for rod-cutting finder opskæring af stang af længde n og ikke kun r_n).

I resten af denne opgave kigger vi på *plate-cutting*-problemet. En input-instans for dette problem består af

1. en rektangulær plade, specificeret ved en heltalsbredde $b \geq 1$ og en heltalshøjde $h \geq 1$,
2. en pris p_{ij} for et delrektangel af bredde i og højde j for alle heltal i og j, hvor $1 \leq i \leq b$ og $1 \leq j \leq h$, og
3. et heltal $s \geq 0$.

Målet er at finde en udskæring af pladen i mindre rektangulære delplader, så den samlede fortjeneste ved at sælge disse plader maksimeres. Saven, der anvendes, kan dog kun foretage udskæringer, der kan opnås via vandrette eller lodrette snit, der hver især opsplitter en rektangulær delplade i to mindre rektangulære delplader. Alle delplader skal have heltalsbredde og heltalshøjde. Da saven efter noget tid bliver sløv, er der yderligere den begrænsning, at summen af længderne af alle snit i udskæringen højst må være s. Ligesom i rod-cutting-problemet er der ikke nogen omkostning ved at lave de enkelte snit.

Et delproblem er specificeret ved heltal i, j og s', hvor $1 \leq i \leq b$, $1 \leq j \leq h$ og $0 \leq s' \leq s$. Lad $r_{i,j,s'}$ være den maksimale fortjeneste, der kan opnås ved udskæring af en delplade med bredde i og højde j, og hvor summen af alle snitlængder i udskæringen af denne delplade højst er s'.

Betragt følgende rekursionsformel for tilfældet, hvor enten $b > 1$ eller $h > 1$:

$$r_{b,h,s} = \max\{p_{b,h}, \max\{r_{i,h,s_1} + r_{b-i,h,s_2} | 1 \leq i < b, s_1 + s_2 \leq s - h\}, \\ \max\{r_{b,j,s_1} + r_{b,h-j,s_2} | 1 \leq j < h, s_1 + s_2 \leq s - b\}\}.$$

Her antages det implicit, at alle parametre i, j, s_1 og s_2 er heltal, at s_1 og s_2 er ikke-negative, samt at max over en tom mængde er $-\infty$.

Del 2.2 (10%) Argumentér for at rekursionsformlen er korrekt og kom herunder ind på optimal delstruktur. Vis desuden at $r_{b,h,s} = p_{b,h}$, når $b = h = 1$.

Løsning til 2.2 Vi viser først optimal delstruktur. Betragt en optimal løsning til problemet svarende til tuplen (b, h, s), og antag at denne indeholder mindst et snit. Antag snittet er vertikalt (horisontale snit er symmetriske), og vælg $1 \leq i < b$, så de to delproblemer har (bredde,højde) hhv. (i, h) og $(b - i, h)$. Da den samlede snitlængde højst er s, og da det vertikale snit har længde h, er den samlede snitlængde for de to delproblemer højst $s - h$. Der findes derfor positive heltal s_1 og s_2, så $s_1 + s_2 \leq s - h$ (faktisk kan man antage $s_1 + s_2 = s - h$), og så de to delproblemer svarer til tuplerne (i, h, s_1) og $(b - i, h, s_2)$. Den optimale løsning til (b, h, s) må indeholde optimale løsninger til disse to delproblemer, for ellers kunne en sub-optimal løsning til et af delproblemerne erstattes med en optimal løsning til det pågældende delproblem, hvilket ville give en bedre løsning til problemet (b, h, s) end den givne optimale løsning, modstrid. Dette viser optimal delstruktur.

Det følger nu, at $r_{b,h,s} = r_{i,h,s_1} + r_{b-i,h,s_2}$ i tilfældet ovenfor. I tilfældet med et horisontalt snit fås tilsvarende $r_{b,h,s} = r_{b,j,s_1} + r_{b,h-j,s_2}$ for passende $1 \leq j < h$ og s_1 og s_2 som ovenfor. Endelig fås $r_{b,h,s} = p_{b,h}$, hvis den optimale løsning til (b, h, s) består af at sælge hele pladen uden opskæringer.

Alle snit, der tages max over i rekursionsformlen, svarer til lovlige løsninger, så der gælder $\geq$ i rekursionsformlen. Af ovenstående følger, at der også gælder $\leq$.

I basistilfældet $b = h = 1$ er ingen opskæring mulig (da delpladerne skal have heltalslængde og -bredde), så derfor er $r_{b,h,s} = p_{b,h}$.

Point for:

1. at forstå, hvilken del af formlen der svarer til lodret/vandret/intet snit,
2. at vise optimal delstruktur (cut-and-paste-argument som i rod-cutting), og hvorfor dette er en del af argumentet for, at formlen er korrekt,
3. at håndtere basistilfældet $b = h = 1$.

Del 2.3 (13%) Beskriv (i ord og/eller pseudo-kode) en dynamisk programmings-algoritme til at finde $r_{b,h,s}$, argumentér for korrektheden af denne og analysér dens køretid og pladsforbrug. Både køretid og plads skal være polynomiel i b og h. Gør det klart, om din algoritme er en bottom-up-algoritme eller en top-down-algoritme med memoization.

Vink: for køretidsanalysen, vis først at den er polynomiel i b, h og s. Vis dernæst, at vi kan antage, at $s = O(bh)$.

Løsning til 2.3 Vi giver en bottom-up-algoritme for problemet. Først initialiseres en tabel med en indgang $r[b', h', s'] \leftarrow p_{b',h'}$ for hvert delproblem svarende til tuplen (b', h', s'). For-løkker itererer over disse tupler i leksikografisk stigende orden (f.eks. yderste for-løkke for b', næstyderste for h' og tredjeyderste for s'). For en given iteration svarende til tuplen (b', h', s') itereres først over alle heltal i, s_1 og s_2, hvor $1 \leq i < b'$, $s_1, s_2 \geq 0$ og $s_1 + s_2 \leq s' - h'$, og opdateringen $r[b', h', s'] \leftarrow \max\{r[b', h', s'], r[i, h', s_1] + r[b' - i, h', s_2]\}$ foretages. Dernæst itereres over alle heltal j, s_1 og s_2, hvor $1 \leq j < h'$, $s_1, s_2 \geq 0$ og $s_1 + s_2 \leq s' - b'$, og opdateringen $r[b', h', s'] \leftarrow \max\{r[b', h', s'], r[b', j, s_1] + r[b', h' - j, s_2]\}$ foretages.

Korrektheden følger af, at algoritmen direkte implementerer rekursionsformlen, og at delproblemerne enumereres i en sådan rækkefølge, at alle del-delproblemer, som et delproblem (b', h', s') afhænger af, er løst, når algoritmen når til (b', h', s').

Vi analyserer dernæst køretiden. Der enumereres over $b \cdot h \cdot s$ tupler, og for hver sådan tupel (b', h', s') tager det $O(b' \cdot (s' - h')^2 + h' \cdot (s' - b')^2) = O((b + h)s^2)$ tid at iterere over alle i, s_1 og s_2 med $1 \leq i < b'$ og $s_1 + s_2 \leq s' - h'$ (og alle j, s_1 og s_2 med $1 \leq j < h'$ og $s_1 + s_2 \leq s' - b'$) for at finde værdien $r[b', h', s']$ (dette kan forbedres, hvis man f.eks. observerer, at vi altid kan vælge $s_2 = s' - h' - s_1$, når s_1 er fastlagt). Den samlede tid over alle tupler (b', h', s') er derfor $O(b \cdot h \cdot (b + h) \cdot s^3)$. Pladsforbruget er domineret af størrelsen af tabellen r, som er $O(b \cdot h \cdot s)$.

Vi kan antage, at $s \leq bh$, idet den maksimale snitlængde mulig for en udskæring må være tilfældet, hvor hele $b \times h$-pladen udskæres i delplader hver af bredde 1 og højde 1 (så faktisk kan vi antage $s \leq (b - 1)(h - 1)$). Af ovenstående fås nu, at køretiden er $O(b^4 \cdot h^4 \cdot (b + h))$, og at pladsforbruget er $O(b^2 h^2)$, som begge er polynomielle i b og h.

Point for:

1. korrekt beskrivelse af algoritme,
2. at forklare, om det er en bottom-up eller top-down med memoization,
3. korrekthedsbevis - at det følger af korrektheden af rekursionsformlen, og at denne implementeres rekursivt (ved top-down), eller at delproblemer løses i rigtig rækkefølge (ved bottom-up),
4. korrekt analyse af tid/plads: at vise, at disse er polynomielle i b, h og s (graden af polynomiet er ikke vigtig),
5. at vise en grænse på tid og plads, der er polynomiel i b og h (graden af polynomiet er ikke vigtig).

Opgave 3 (30%)

Denne opgave omhandler paradigmet Grådige Algoritmer.

Del 3.1 (8%) Giv en overordnet beskrivelse af dette paradigme og kom ind på det, du vurderer er det vigtigste inden for emnet Grådige Algoritmer. Brug gerne figurer og undlad beviser. Eventuelle håndkørsler skal være på små instanser. Teksten i din besvarelse skal fylde en halv til en hel side; der er ingen grænse på, hvor meget figurer må fylde.

Løsning til 3.1 Point gives f.eks. for:

1. overordnet forklaring af, hvad en grådig algoritme/et grådigt valg er, hvor der evt. kommes ind på forskelle mellem grådige algoritmer og DP-algoritmer (lav grådigt valg, før delproblemer løses),
2. at nævne, at der kun er et delproblem tilbage efter det grådige valg,
3. at forklare Greedy Choice Property i ord, evt. med eksempel fra bogen (activity selection eller Huffman),
4. at forklare Optimal Delstruktur i ord, evt. med eksempel fra bogen (activity selection eller Huffman),
5. at nævne at GCP+OSS giver korrekthed af grådig algoritme,
6. at håndkøre enten algoritmen for Huffman eller for activity selection
7. at komme ind på køretider for Huffman eller activity selection.

I resten af opgaven betragter vi en variant af strengkomprimerings-problemet fra afsnit 16.3 i kursusbogen. I stedet for at komprimere en tekststreng til en binær streng, kigger vi her i stedet på problemet med at komprimere en tekststreng til en streng bestående af symboler fra mængden $\Sigma = \{0, 1, 2\}$ vha. prefix codes. Som i kursusbogen er input en liste af n heltal, der specificerer frekvenserne for hvert af de n typer af bogstaver i strengen, der skal komprimeres [1]. Vi ønsker en komprimeret streng af minimal længde bestående af symboler fra Σ.

For at simplicifere opgaven antager vi, at n er ulige og mindst 3. Der gælder da, at i ethvert optimalt parse-træ for input har hver indre knude (knude der ikke er et blad) netop tre børn. Dette resultat må gerne benyttes i det følgende og skal ikke bevises.

I det følgende kigger vi på en modificeret version af Huffmans algoritme, der finder et optimalt parse-træ, når mængden af symboler er Σ. I stedet for at vælge to symboler i hvert skridt (som i kursusbogen) vælger den modificerede version tre symboler med de mindste frekvenser og tilføjer dem som børn af en fælles forælder, der har frekvens lig med summen af børnenes frekvenser. Ellers fungerer algoritmen på samme måde som beskrevet i kursusbogen.

Del 3.2 (6%) Håndkør den modificerede Huffman-algoritme på instansen bestående af de $n = 7$ bogstaver a til g, hvor $a.freq = 2$, $b.freq = 3$, $c.freq = 4$, $d.freq = 5$, $e.freq = 6$, $f.freq = 7$ og $g.freq = 8$. Hvert skridt i håndkørslen skal illustreres, og den resulterende prefix-kode for hvert bogstav skal angives.

[1] Som i kursusbogen tillader vi n at være vilkårlig stor, så strengen, der skal komprimeres, kan bestå af andet end blot bogstaverne a til å.

Løsning til 3.2 Først sammensættes a, b og c ved at give dem en fælles forælder z med frekvens $2+3+4=9$. Dernæst sammensættes d, e og f ved at give dem en fælles forælder z' med frekvens $5+6+7=18$. Til sidst sammensættes g, z og z' med en fælles forælder r med frekvens $8+9+18=35$.

Point for:

1. korrekt håndkørsel med illustrationer (vejledende løsning ovenfor er ikke tilstrækkelig for max point),
2. beskrivelse i ord af, hvad algoritmen gør i hvert skridt (f.eks. som i vejledende løsning, evt. med detaljer om, hvilke prioritetskø-operationer der udføres).

Del 3.3 (6%) Analysér køretiden for den modificerede Huffman-algoritme.

Løsning til 3.3 I hver iteration foretages tre ExtractMin-operationer og en INSERT-operation i algoritmens prioritetskø, hvilket dominerer tiden for denne iteration. Ved brug af en almindelig binær min-heap som prioritetskø er tiden $O(\log n)$ per iteration. Da antallet af elementer i køen i reduceres med 2 i hver iteration (tre elementer fjernes og et indsættes), er antallet af iterationer $O(n)$. Den samlede køretid for algoritmen er derfor $O(n \log n)$.

Point for:

1. at nævne, at der benyttes en min-heap, hvor nøgler er frekvenser,
2. at forklare, hvilke min-heap-operationer, der udføres i hver iteration,
3. køretiden for disse operationer i en iteration,
4. korrekt argument for, at antal iterationer er $O(n)$ (reduktion med to elementer i hver iteration),
5. at give den korrekte samlede køretid.

Del 3.4 (2%) I kursusbogen defineres omkostningen $B(T)$ af et parse-træ T, og denne omkostning er netop antallet af bits i den tilhørende komprimerede bitstreng. Kan vi definere $B(T)$ på samme måde, når vi betragter mængden af symboler Σ, så $B(T)$ ligeledes udtrykker længden af den komprimerede streng? Argumentér for dit svar.

Løsning til 3.4 Ja. Benytter vi samme definition, gælder $B(T) = \sum_{c \in C} c.freq \cdot d_T(c)$. Her er $d_T(c)$ netop antallet af symboler fra Σ, der bruges i prefix-koden for en enkelt forekomst af c. Det samlede antal symboler, der bruges til at repræsentere forekomster af c i tekststrengen er derfor $c.freq \cdot d_T(c)$. Altså er $B(T)$ netop længden af den komprimerede streng af symboler fra Σ.

Point for:

1. at kunne svare ja,
2. at kunne argumentere for dette svar som ovenfor.

Del 3.5 (8%) Bevis at den modificerede algoritme har Greedy Choice Property ved at tilpasse beviset for Greedy Choice Property for Huffmans algoritme i kursusbogen. Husk at nævne, hvad det grådige valg for den modificerede algoritme er. Optimal Substructure skal ikke bevises.

Løsning til 3.5 Det grådige valg er at vælge tre elementer i prioritetskøen med minimale frekvenser og sammensætte dem ved at give dem en fælles forælder.

For at vise GCP, skal vi vise, at der findes en optimal løsning, der indeholder det grådige valg. Betragt et optimal parse-træ T, og antag at det ikke indeholder det grådige valg (ellers er vi færdige). Vi vil vise, at vi kan transformere T til et andet optimalt parse-træ, som indeholder det grådige valg. Vælg tre søskendeblade a, b og c med maksimal dybde i T, hvor $a.freq \leq b.freq \leq c.freq$. Sådanne søskendeblade findes grundet resultatet ovenfor, at alle indre knuder i T har netop tre børn.

Lad x, y og z være bladene i T, som den modificerede Huffman-algoritme ville have valgt i første skridt, hvor $x.freq \leq y.freq \leq z.freq$. Vi viser, at a og x kan ombyttes, uden at B-omkostningen øges. Lad T' være T med disse to blade ombyttet. Da gælder åbenlyst $B(T') \geq B(T)$, da T er optimalt. Desuden fås

$$\begin{aligned} B(T') &= B(T) - d_T(x) \cdot x.freq - d_T(a) \cdot a.freq + d_T(x) \cdot a.freq + d_T(a) \cdot x.freq \\ &= B(T) + (d_T(x) - d_T(a))(a.freq - x.freq). \end{aligned}$$

Da a har maksimal dybde i T, gælder $d_T(x) - d_T(a) \leq 0$, og da x har minimal frekvens i T, gælder $a.freq - x.freq \geq 0$. Dermed fås

$$B(T') = B(T) + (d_T(x) - d_T(a))(a.freq - x.freq) \leq B(T) + 0 = B(T).$$

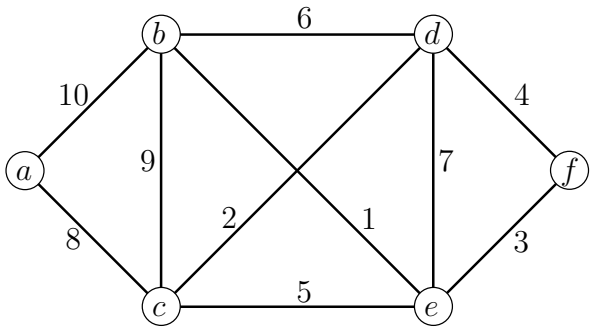
Dermed er $B(T') = B(T)$, så T' er optimalt. Tilsvarende kan vi ombytte y og b og z og c, uden at omkostningen af træet ændres. Dette viser GCP: der findes en optimal løsning, der indeholder det grådige valg.

Point for:

1. at forklare, hvad det grådige valg er (tre elementer med minimale frekvenser får fælles forælder),
2. at forklare, hvad der skal vises ved GCP (at der findes en optimal løsning, der indeholder det grådige valg),
3. at kunne forklare, hvilke blade i en optimal løsning, der skal ombyttes for at vise GCP (3 blade med minimale frekvenser ombyttes med 3 søskende-blade af maksimal dybde),
4. at kunne forklare intuitionen ved, at en ombytning ikke giver en dårligere løsning eller (fuld point) at kunne lave udregningen, som i den vejledende løsning (en ombytning er tilstrækkelig at analysere).

Opgave 4 (24%)

Denne opgave omhandler minimum udspændende træ-problemet (Minimum Spanning Tree, forkortet MST).



Figur 1: Grafen H med vægte på kanterne.

Lad H være grafen i Figur 1. For hver kant er der angivet en vægt.

Del 4.1 (6%) Vis forløbet af Kruskals algoritme, når vi ønsker at finde et minimum udspændende træ i H. For hvert skridt skal der vises den del af en minimum udspændede skov der er fundet indtil nu. Desuden skal der vises, hvorledes det besluttes, om den aktuelt betragtede kant skal tages med i det endelige træ. Det er ikke nødvendigt at tegne alle kanterne med vægte for hvert skridt. Man skal blot vise de kanter der indtil nu er inkluderet i den aktuelle skov.

Løsning til 4.1 (skitse) Kanterne (b, e), (c, d), (e, f), (d, f) og (a, c) indsættes i skoven i denne rækkefølge.

Point for:

1. korrekt håndkørsel med illustrationer (vejledende løsning ovenfor er ikke tilstrækkelig),
2. korrekt gennemgang af, hvorledes det i hvert skridt besluttes, om den aktuelle kant skal med i træet eller ikke.

I de to næste delopgaver betegner T det MST i H, der består af kanterne (a, c), (c, d), (b, e), (d, f) og (e, f).

Del 4.2 (2%) Argumentér for, at hvis vi ændrer vægten af kanten (d, f) til 6, så vil T ikke længere være et MST i H.

Løsning til 4.2 T er ikke længere et MST i H, idet vi kan fjerne (d, f) fra T og genforbinde de to opnåede deltræer med den billigere kant (c, e) (alternativ/mere besværlig løsning: kør Kruskal med de nye vægte og observér, at det fundne MST har vægt (ift. de nye kantvægte) skarpt mindre end T).

Point for:

1. korrekt argument som ovenfor (at køre algoritmen forfra er mere besværligt, men skal stadig give fuld point, hvis argumentet er korrekt).

Del 4.3 (2%) Argumentér for, at hvis vi ændrer vægten af kanten (b, c) til 3, så vil T ikke længere være et MST i H (vægten af (d, f) er her 4 som i figuren).

Løsning til 4.3 T er ikke længere et MST i H, idet vi kan fjerne (d, f) fra T og genforbinde de to opnåede deltræer med den billigere kant (b, c) (alternativ løsning: kør Kruskal med de nye vægte og observér, at det fundne MST har vægt (ift. de nye kantvægte) skarpt mindre end T).

Point for:

1. korrekt argument som ovenfor (at køre algoritmen forfra er mere besværligt, men skal stadig give fuld point, hvis argumentet er korrekt).

(*Eksamenssættet fortsætter på næste side*)

Nedenstående to spørgsmål relaterer sig til et fast MST M i en graf $G = (V, E, w)$ med kantvægte givet ved vægt-funktion w. Lad x være en fast kant i E og lad $k \in \mathbb{N}$ være et fast positivt heltal.

Lad $G' = (V, E, w')$, hvor w' er identisk med w for alle kanter i $E \setminus \{x\}$, og hvor $w'(x) = k$.

Del 4.4 (7%) Antag at x tilhører M. Beskriv en egenskab ved G (herunder en egenskab ved vægtfunktionen w), så

1. hvis M er et MST i G', da må denne egenskab gælde, og
2. hvis egenskaben gælder, da er M et MST i G'.

Argumentér for dine svar.

Vink: Se på M minus kanten x og en passende valgt delmængde af E. Hvad må der gælde i G' om vægten $w'(x) = k$ ift. vægtene af kanterne i denne delmængde? For at vise del 2, argumentér for, at egenskaben sikrer, at den generiske MST-algoritme i kursusbogen kan finde M i G'.

Løsning til 4.4 Betragt snittet (V_1, V_2), hvor V_1 og V_2 er knudemængderne for de to træer, der opnås ved at fjerne x fra M. Vi vil vise, at vi kan benytte følgende egenskab: x er en let kant i G', der krydser dette snit, eller ækvivalent: alle kanter over snittet i G' har vægt mindst k.

Antag først, at M er et MST i G'. Vi skal vise, at da gælder egenskaben. Antag at det modsatte var tilfældet. Da findes en anden kant y, der krydser (V_1, V_2), og som har skarpt mindre vægt end x i G'. Fjernes x fra M, og genforbindes de to opnåede deltræer med y fås et billigere udspændende træ i G' end M, modstrid. Altså gælder egenskaben.

Antag nu at egenskaben gælder. Vi påstår, at det for enhver kant y i M gælder, at y er en let kant i G', der krydser snittet (Y_1, Y_2) defineret ved de to deltræer i $M \setminus y$. Dette følger af den antagede egenskab, hvis $y = x$, og hvis $y \neq x$, følger det af, at alle kanter, der krydser (Y_1, Y_2), har samme vægt i G og i G', og at y er en let kant for snittet i G.

Betragt nu den generiske algoritme i kursusbogen anvendt på G'. Vælges kanterne for denne til at være netop kanterne i M (i en vilkårlig rækkefølge), da vil algoritmen tilføje netop disse kanter til det MST, der opbygges grundet ovenstående og grundet Theorem 23.1 i kursusbogen. Altså er M et MST i G'.

Point for:

1. at kunne nævne, hvilket snit der betragtes,
2. at kunne give den korrekte egenskab,
3. at kunne argumentere korrekt for implikation nr. 1,
4. at kunne argumentere korrekt for implikation nr. 2.

Del 4.5 (7%) Antag nu at x ikke tilhører M. Beskriv en egenskab ved G (herunder en egenskab ved vægtfunktionen w), så

1. hvis M er et MST i G', da må denne egenskab gælde, og
2. hvis egenskaben gælder, da er M et MST i G'.

Argumentér for dine svar.

Vink: se på vejen mellem de to endepunkter af x i M. Hvad må der gælde i G' om vægten $w'(x) = k$ ift. vægten af kanterne på denne vej? For at vise del 2, argumentér (ligesom i den tidligere delopgave) for, at egenskaben sikrer, at den generiske MST-algoritme i kursusbogen kan finde M i G'.

Løsning til 4.5 Lad P være vejen i M mellem endepunkterne for x. Vi vil vise, at vi kan benytte følgende egenskab: vægten af x i G' er mindst lige så stor som alle vægte af kanterne på P. Ækvivalent: alle kanter på P har vægt højst k.

Antag først at M er et MST i G'. Da må egenskaben gælde, for ellers findes der en kant y på P, som har skarpt større vægt end x i G'; fjernes y fra P, og genforbindes de to opnåede træer med x, fås et udspændende træ i G' af vægt skarpt mindre end M, modstrid.

Antag nu, at egenskaben gælder. Da påstår vi, at der for enhver kant y i M gælder, at y er en let kant i G', der krydser snittet (Y_1, Y_2) defineret ved de to deltræer i $M \setminus y$. Bemærk at y er en let kant for dette snit i G, så i G' er vægten af y højst vægten af enhver kant $z \neq x$, der krydser (Y_1, Y_2). Hvis x ikke krydser dette snit, er y derfor en let kant for snittet i G'. Hvis x krydser snittet, må $y \in P$, men så følger det af den antagede egenskab ovenfor, at vægten af y højst er vægten af x i G'. I alle tilfælde gælder altså, at y er en let kant i G', der krydser snittet (Y_1, Y_2). Da dette gælder for alle $y \in M$, giver det samme argument som for den tidligere delopgave, at den generiske algoritme anvendt på G' kan opnå M, så M er et MST i G'.

Point for:

1. at kunne give den korrekte egenskab,
2. at kunne argumentere korrekt for implikation nr. 1,
3. at kunne argumentere korrekt for implikation nr. 2.

EKSAMENSSÆTTET SLUTTER HER