Danmarks Tekniske Universitet

Skriftlig prøve, den 16. maj 2012.

Kursusnavn: Algoritmer og datastrukturer I Kursus nr. 02105.

Tilladte hjælpemidler: Skriftlige hjælpemidler.

Varighed: 4 timer

Vægtning af opgaverne: Opgave 1 - 24%, Opgave 2 - 16%, Opgave 3 - 20%, Opgave 4 - 20%, Opgave 5 - 20 %.

Vægtningen er kun en cirka vægtning.

Alle opgaver besvares ved at udfylde de indrettede felter nedenfor. Som opgavebesvarelse afleveres blot denne og de efterfølgende sider i udfyldt stand. Hvis der opstår pladsmangel kan man eventuelt benytte ekstra papir som så vedlægges opgavebesvarelsen.

Opgave 1 (kompleksitet)

1.1 Angiv for hver af nedenstående udsagn om de er korrekte:

	Ja	*Nej*
$\frac{1}{20}n^2 + 100n^3 = O(n^2)$	☐	☒
$(\log n)^2 + n = O(n)$	☒	☐
$(\log n)^2 + n = \Theta(n)$	☒	☐
$2^{\log_2 n} = O(n)$	☒	☐
$n^3(n-1)/5 = \Theta(n^3)$	☐	☒

1.2 Skriv følgende liste af funktioner op i voksende rækkefølge efter asymptotisk vækst. Dvs. hvis funktionen $g(n)$ følger umiddelbart efter funktionen $f(n)$ i din liste, så skal der gælde at $f(n) = O(g(n))$.

$n(\log n)^2$

$n^2 \log n$

$n^3 + 100n - 5000n^2$

2^4

$n^{1/4}$

2^n

Svar: $2^4, n^{1/4}, n(\log n)^2, n^2 \log n, n^3 + 100n - 5000n^2, 2^n$

1.3 Antag at du har en algoritme hvis køretid er præcist $3n^4$. Hvor meget langsommere kører algoritmen hvis du fordobler inputstørrelsen?

A dobbelt så langsom	B 3 gange langsommere	C 4 gange langsommere
D 8 gange langsommere	E 12 gange langsommere	**F** 16 gange langsommere

1.4 Betragt nedenstående algoritme.

Algorithm 1 Løkke1(n)

```
1: for i = 1 to n do
2:     j = 1
3:     while j ≤ n/2 do
4:         print "⋆"
5:         j = j + 1
6:     end while
7: end for
```

Køretiden af algoritmen er

A $\Theta(\log n)$	B $\Theta(n)$	C $\Theta(n \log n)$	D $\Theta(n^2 \log n)$	E $\Theta(n^3)$
F $\Theta(n^2)$	G $\Theta(2^n)$	H $\Theta(n^4)$	I $\Theta(\sqrt{n})$	

1.5 Betragt nedenstående algoritme.

Algorithm 2 Løkke2(n)

```
1: for i = 1 to n do
2:   for j = 1 to n do
3:     for k = j to n do
4:       print "⋆"
5:     end for
6:   end for
7: end for
```

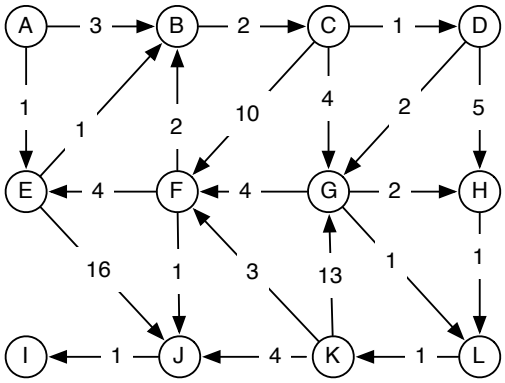
Køretiden af algoritmen er

[A] $\Theta(\log n)$ [B] $\Theta(n)$ [C] $\Theta(n \log n)$ [D] $\Theta(n^2 \log n)$ [**E**] $\Theta(n^3)$

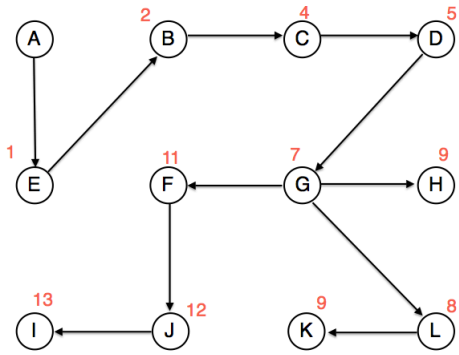
[F] $\Theta(n^2)$ [G] $\Theta(2^n)$ [H] $\Theta(n^4)$ [I] $\Theta(\sqrt{n})$

Opgave 2 (grafer)

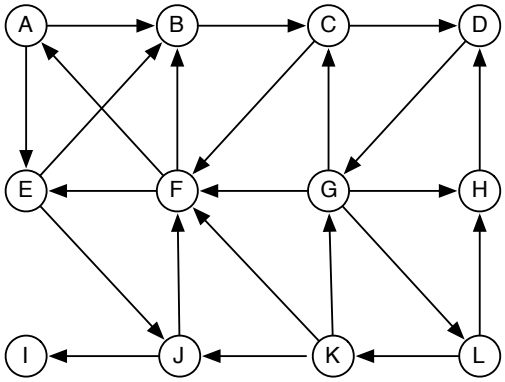
2.1 Angiv et korteste veje træ for nedenstående graf når korteste veje beregningen sker med hensyn til startknuden A. Angiv for hver knude afstanden fra knuden A.



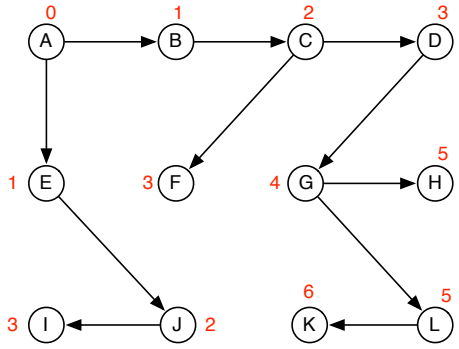
Angiv korteste veje træet og afstandene her:



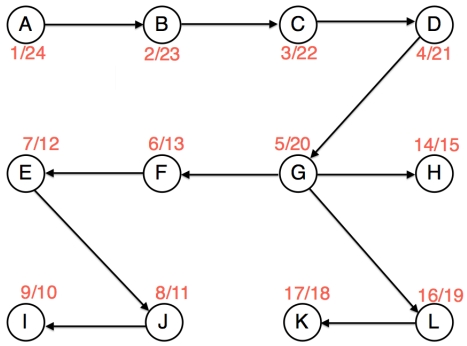
2.2 Betragt nedenstående graf G.



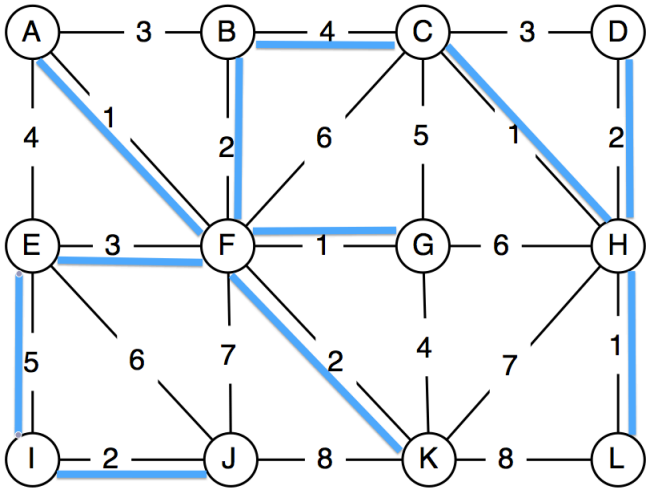
a) Angiv et BFS træ for grafen G når BFS gennemløbet starter i knuden A. Angiv BFS-dybde/lag for hver knude. Det antages at incidenslisterne er sorteret i alfabetisk orden.



b) Angiv et DFS træ for grafen G, når DFS gennemløbet starter i knuden A. Angiv en DFS nummerering af knuderne. Det antages at incidenslisterne er sorteret i alfabetisk orden.



2.3 Angiv et mindste udspændende træ i nedenstående graf.



Opgave 3 (modellering og anvendelse af algoritmer/datastrukturer)

Kabeltv-firmaet AlgoNet udbyder kabeltv til alle huse i den lille by AlgoCity. Firmaet sender kabeltv fra deres hovedstation i byen ud til alle de huse der ønsker det. De har et netværk af kabler de bruger til at sende tv-signal til alle kunderne i byen. Kablerne går mellem nogle bokse. Der er en boks i alle de huse der modtager kabeltv (dvs. hos alle kunderne), og ingen bokse andre steder. Der kan godt være mange kabler tilsluttet samme boks, så der kan sendes signal både ud og ind af en boks. Der er X huse og K kabler i netværket. Firmaet kender også den præcise længde af hvert kabel.

Opgave 3.1: Rutning

Firmaet ønsker at finde ud af hvad vej de skal sende signalerne fra hovedstationen til de huse der ønsker kabeltv. Der skal være så lidt forringelse af signalet som muligt, så derfor ønsker de at hver kunde skal modtage signalet af så kort en rute som muligt (længden af en rute er den samlede længde af kablerne på ruten).

Giv en algoritme der finder den bedste måde at rute signalerne på, så hver kundes signal er så godt som muligt. Angiv køretiden af din algoritme og argumenter for at den er korrekt.

Svar Modeller problemet som en graf. Hvert hus har præcis en boks, og modelleres som en knude i grafen. Hovedstationen modelleres også som en knude i grafen. Hvert kabel forbinder to bokse (eller en boks og hovedstationen), og modelleres som en ikke-orienteret kant mellem de to tilsvarende knuder i grafen med en kantlængde lig kabelets længde. Den færdige graf har altså $O(X)$ knuder og $O(K)$ kanter.

Den korteste rute fra hovedstationen til enhver boks kan nu findes korrekt ved at anvende Dijkstras Algoritme, eftersom algoritmen garanterer at den finder den korteste sti fra en knude til alle andre knuder. Køretiden er $O(X + K \log X)$ (se slides) hvis grafen er implementeret med incidenslister og en prioritetskø implementeret med en hob. Kanterne i korteste vej træet som Dijkstras Algoritme producerer er de kanter som signalet fra hovedstationen skal rutes ad.

Opgave 3.2: Forringelse af signal i bokse

Firmaet har fundet ud af at der også sker en forringelse af signalet, når det går gennem en boks. Forringelsen af signalet når det går gennem en boks, svarer til den forringelse der sker når det løber gennem 5 meter kabel.

Giv en algoritme der finder den bedste måde at rute signalerner på, så hver kundes signal er så godt som muligt. Angiv køretiden af din algoritme og argumenter for at den er korrekt.

Svar Forringelsen af signalet sker når det går igennem en boks (og ikke når det går ind i en boks). Vi kan modellere forringelserne i boksene som en 5 meter forlængelse af de udgående kabler fra hver boks (idet det er så meget signalet bliver forringet ved at gå igennem en boks). Ved at løse problemet med disse nye kantlængder giver Dijkstras Algoritme et nyt korteste vej træ der tager højde for forringelsen i boksene.

For at kunne forlænge udgående kanter er det nødvendigt at modellere hver kabel som to orienterede kanter i stedet for en enkelt ikke-orienteret kant. Alle udgående kanter fra en boks tildeles en kantlængde der er 5 meter længere end kablets længde. Herefter anvendes Dijkstras Algoritme som tidligere fra hovedstationen. Køretiden er som før $O(X + K \log X)$ fordi antallet af kanter i grafen højst fordobles.

Opgave 3.3: Vedligeholdelse

Firmaet skal spare, og ønsker derfor ikke længere at vedligeholde hele kabelnettet. De vil derfor gerne finde en mængde af kabler der er billigst mulig at vedligeholde, men som samtidig forbinder alle nuværende kunder. Prisen for at vedligeholde et kabel er proportional med længden af kablet, dvs. det er den samlede længde af kablerne der afgør hvor dyrt det er at vedligeholde dem.

Giv en algoritme der finder den billigste mængde af kabler at vedligeholde. Angiv køretiden af din algoritme og argumenter for at den er korrekt.

Svar Betragt som før problemet som en graf hvor husene og hovedstationen er knuder og hvert kabel modelleres som en ikke-orienteret kant i grafen med en kantlængde svarende til kablets længde. Firmaet ønsker nu at minimere den samlede længde af kabler.

Problemet kan løses ved at finde det mindste udspændede træ i grafen, hvor vægten af en kant er lig med kantlængden (og dermed kablets længde). Det mindste udspændende træ er netop det træ der forbinder alle knuder (så alle kunder er forbundet), og hvor summen af kantvægtene er minimeret (svarende til at den samlede længde af kabler er kortest mulig). Hvis kun de kanter der er med i det mindste udspændende træ vedligholdes minimeres vedligeholdessesudgiften dermed.

Hvis det mindste udspændende træ findes med Prims Algoritme er køretiden $O(K \log X)$ hvis grafen er implementeret med incidenslister og en prioritetskø implementeret med en hob.

Opgave 4 (Træer)

Denne opgave omhandler rodfæstede binære træer. Hver knude har *enten to eller ingen* børn. Knuden x's venstre barn betegnes $left[x]$, og dens højre barn betegnes $right[x]$. Hvis knuden x ikke har nogle børn, har $left[x]$ og $right[x]$ den specielle NIL værdi. Hvis knude x ikke har nogle børn, kaldes den et blad. Ellers kaldes den en intern knude. Såfremt rodknuden for et træ er NIL, er træet tomt. Til hver knude i træet er der knyttet en vægt; knuden x har vægten $weight[x]$.

4.1 Ens vægte

Giv en effektiv algoritme EnsVægt(x) der givet rodknuden til et vægtet binært træ returnerer 1 hvis der er mindst to knuder i træet der har samme vægt og 0 ellers.

Angiv køretiden af din algoritme og argumenter for at den er korrekt.

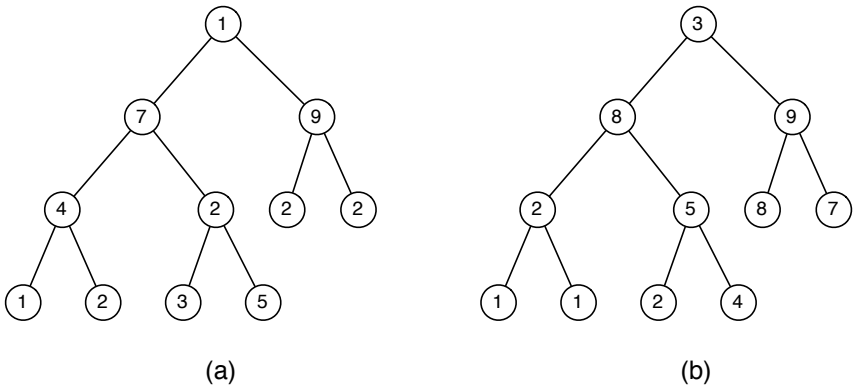
Svar Lad algoritmen gennemløbe alle knuder i træet en gang ved brug af DFS. Vedligehold desuden en datastruktur D der supporterer Search og Insert operationerne over knudernes vægte. For hver knude x der besøges i træet tjekkes først om vægten allerede findes i D med Search($weight[x]$). Hvis vægten findes returnerer algoritmen 1. Ellers indsættes vægten i D med Insert($weight[x]$). Når gennemløbet af knuderne er afsluttet returneres 0.

Algoritmen returnerer 1 første gang en knude tjekkes hvor en knude med samme vægt tidligere er blevet tjekket. Hvis der ikke er sådanne kollisioner returneres 0. Køretiden for DFS er $O(|V|)$ (hvor V er sættet af knuder i træet), og køretiden for Search og Insert operationerne er afhængig af hvilken datastruktur D der vælges. Bemærk at der laves $O(|V|)$ Search og Insert operationer. Anvendes som D et balanceret binært søgetræ tager D-operationerne $O(\log |V|)$ tid per styk, så den samlede køretid er $O(|V| \log |V|)$ i worst case. Anvendes i stedet som D en dynamisk hægtet hashtabel med en universel hashfunktion tager D-operationerne $O(1)$ forventet tid per styk, så den samlede forventede køretid for algoritmen er $O(|V|)$.

4.2 Vægtbalancerede træer

To knuder i et træ er søskende, hvis de har samme forælder. Dvs. $left[x]$ og $right[x]$ er søskende. Vi siger at en knude v i et vægtet binært træ er *vægtbalanceret* hvis vægten af v højst er dobbelt så meget som vægten af dens søskende. Et træ er vægtbalanceret hvis alle knuder i træet er vægtbalancerede.

4.2.1 Hvilke af nedenstående træer er vægtbalancerede?



Svar (a) er vægtbalanceret. I (b) findes et søskendepar med vægt 2 og 5, som strider mod definitionen.

4.2.2 Skriv pseudokode for en *rekursiv* algoritme VægtBalance(x), der givet rodknuden til et vægtet binært træ der returnerer 1 hvis træet er vægtbalanceret og 0 ellers.

Angiv køretiden af din algoritme og argumenter for at den er korrekt.

Svar Algoritmen lader en knude x bestemme om børnene er vægtbalancerede. Eftersom et blad ikke har nogen børn er de altid vægtbalancerede. I alle andre tilfælde sammenlignes børnenes vægt, og der returneres 0 hvis det ene barn vejer mere end dobbelt så meget som det andet. Er det ikke tilfældet køres algoritmen rekursivt på børnene, og den mindste returværdi fra disse returneres. Da der kun returneres 0 eller 1 vil det altid være en af disse værdier der returneres fra algorimen, og da den mindste værdi returneres i de rekursive kald vil 0 blive returneret netop hvis der findes et ikke-vægtbalanceret søskendepar.

Fordi hver knude kun tjekkes en gang og der kun udføres et konstant antal sammenligninger per tjek er køretiden af algoritmen $O(|V|)$, hvor V er antallet af knuder i træet.

Algorithm 3 VægtBalance(x)

```
1: if (left[x] = NIL) then
2:    return 1
3: else
4:    if (max{weight[left[x]], weight[right[x]]} > 2 * min{weight[left[x]], weight[right[x]]}) then
5:       return 0
6:    else
7:       return min{VægtBalance(left[x]), VægtBalance(right[x])}
8:    end if
9: end if
```

Opgave 5 (datastrukturer)

5.1 Lad K være en kø. Udfør følgende operationer fra venstre til højre: et bogstav i betyder *Enqueue(K,i)* og $*$ betyder *Dequeue(K)*.

D * T U * * I N * F O R * M * A T I K

Angiv sekvensen af bogstaver der bliver "dequeue"et (returneret af *Dequeue(K)*) af disse operationer:

A D U T I R M

B D U T N R M K I T A O F

C D U T N R M

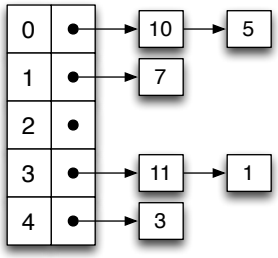
D D T U I N F

E D U T N R M I T A O F

F D U T N O M

5.2 Lad H være en hægtet hashtabel (chained hashing) af størrelse 5 med hashfunktion $h(x) = 3x \mod 5$. Angiv hvordan hashtabellen H ser ud efter indsættelse af tallene 5, 1, 11, 7, 10, 3.

Svar Hashtabellen ser ud som følger. Venstre række i tabellen angiver indexet. Pilene er pointere til det næste element i den hægtede liste. Hvis der ikke er en pil videre fra et felt betyder det at at pointeren er NIL.

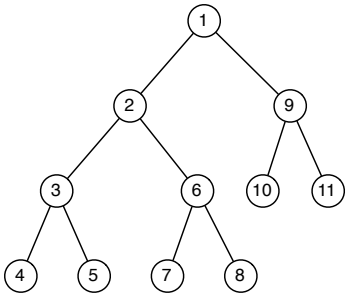


5.3 Lad H være en hashtabel med linær probering (linear probing) af størrelse 5 med hashfunktion $h(x) = x \mod 5$. Angiv hvordan hashtabellen H ser ud efter indsættelse af tallene 6, 2, 7, 16.

Svar Hashtabellen ser ud som følger. Øverste række i tabellen angiver indexet, nederste angiver tallene der er indsat. Bemærk at der ikke er noget element på index 0.

0	1	2	3	4
	6	2	7	16

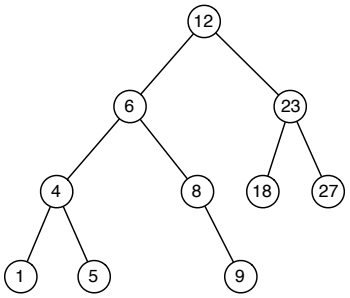
Opgave 5.4 Angiv den rækkefølge knuderne bliver skrevet ud i, når man laver et inorder gennemløb af nedenstående træ.



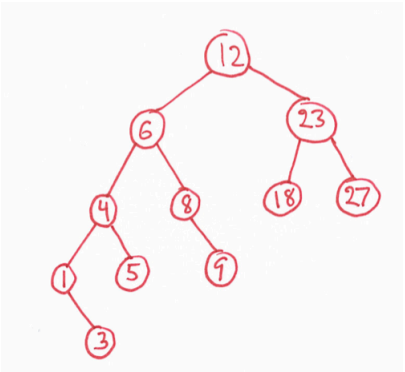
Inorder gennemløb: $4, 3, 5, 2, 7, 6, 8, 1, 10, 9, 11$

5.5 Denne opgave omhandler (ubalancerede) binære søgetræer, som beskrevet i de udleverede noter CLRS kapitel 12.

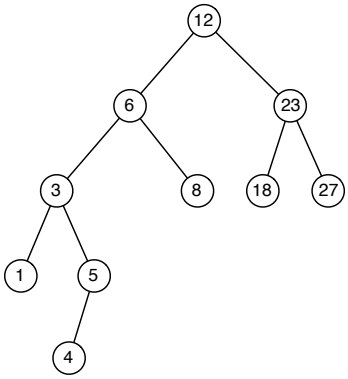
Opgave a Angiv hvordan det binære søgetræ nedenfor ser ud efter indsættelse af et element med nøgle 3.



Svar



Opgave b Angiv hvordan det binære søgetræ nedenfor ser ud efter sletning elementet med nøgle 6.



Svar