Danmarks Tekniske Universitet

Skriftlig prøve, den 29. maj 2013.

Kursusnavn: Algoritmer og datastrukturer Kursus nr. 02326.

Hjælpemidler: Skriftlige hjælpemidler. Det er **ikke** tilladt at medbringe lommeregner.

Varighed: 4 timer

Vægtning af opgaverne: Opgave 1 - 20%, Opgave 2 - 20%, Opgave 3 - 20%, Opgave 4 - 15%, Opgave 5 - 25%.

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.

Vejledende løsning

Opgave 1 (kompleksitet)

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

	Ja	*Nej*
$n(n+1)/100 = O(n^3)$	[x]	☐
$n(n+1)/2 + \log n = O(n^2)$	[x]	☐
$\frac{n}{\log n} = O(n)$	[x]	☐
$\sqrt{n} = O(\log n)$	☐	[x]
$n + n(n-1)/5 = O(n)$	☐	[x]

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 $f(n) = O(g(n))$.

$\frac{n^3}{10000}$

$\frac{n^3}{\log n} + 100n + 5000n^2$

$2^n \log n$

$2^4 \cdot n^{1/4}$

$2n^2 + \sqrt{n}$

Svar: $2^4 \cdot n^{1/4}, \quad 2n^2 + \sqrt{n}, \quad \frac{n^3}{\log n} + 100n + 5000n^2, \quad \frac{n^3}{10000}, \quad 2^n \log n$

1.3 Betragt nedenstående algoritme, der som input tager et array med n heltal.

Algorithm 1 Løkke1(A[1 . . . n])

```
1: for i = 1 to n − 1 do
2:    for j = i + 1 to n do
3:       if A[i]=A[j] return false
4:    end for
5: end for
6: Return true
```

Forklar hvad algoritmen beregner/gør:

Algoritmen returnerer true hvis alle tal i rækken er forskellige og false ellers.

Køretiden af algoritmen er (sæt kun ét kryds): Dit svar skal være så tæt som muligt

[A] $\Theta(\log n)$ [B] $\Theta(n)$ [C] $\Theta(n \log n)$ [D] $\Theta(n^2 \log n)$ [E] $\Theta(n^{1.5})$

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

1.4 Antag at du har en algoritme hvis køretid er præcist $7n^3$. Hvor meget langsommere kører algoritmen hvis du fordobler inputstørrelsen?

A dobbelt så langsom | B 3 gange langsommere | C 7 gange langsommere

D 8 gange langsommere | E 14 gange langsommere | F 4 gange langsommere

1.5 Betragt nedenstående algoritme.

Algorithm 2 Løkke2(n)

```
1: if n > 0 then
2:   for i = 1 to n do
3:     print "⋆"
4:   end for
5:   Løkke2(n − 1)
6: end if
```

a) Hvor mange stjerner udskriver algoritmen Løkke2 når den bliver kaldt med parameteren $n = 4$: **10**

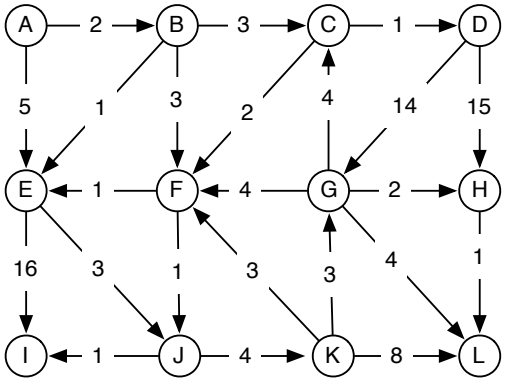
b) Køretiden af algoritmen er (sæt kun ét kryds): Dit svar skal være så tæt som muligt

A $\Theta(\log n)$ | B $\Theta(n)$ | C $\Theta(n \log n)$ | D $\Theta(n^2 \log n)$ | E $\Theta(n^3)$

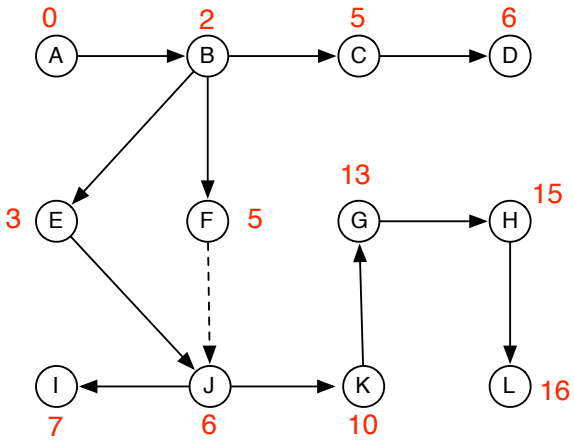
F $\Theta(n^2)$ | G $\Theta(2^n)$ | H $\Theta(n^4)$ | I $\Theta(\sqrt{n})$

Opgave 2 (grafer)

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

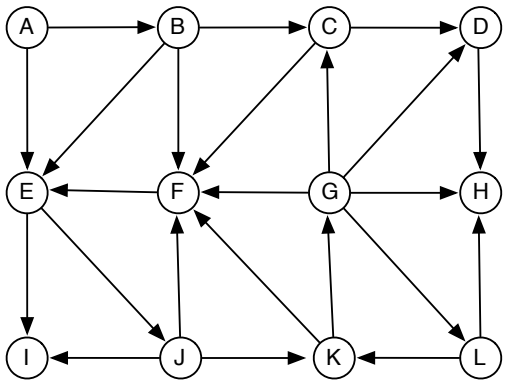


Angiv korteste veje træet og afstandene her:

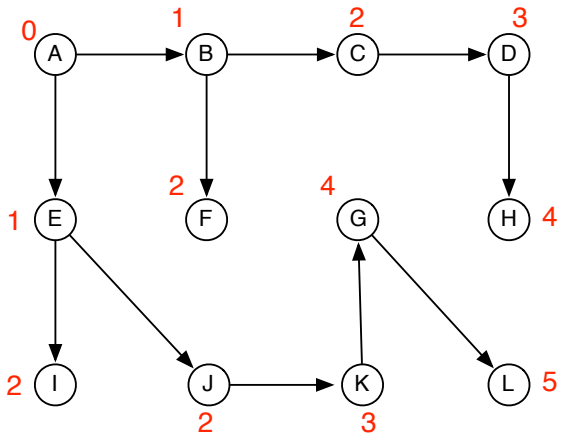


Den stiplede kant er ikke med i i korteste veje træet. Men kanten fra E til J kan erstattes med den stiplede kant.

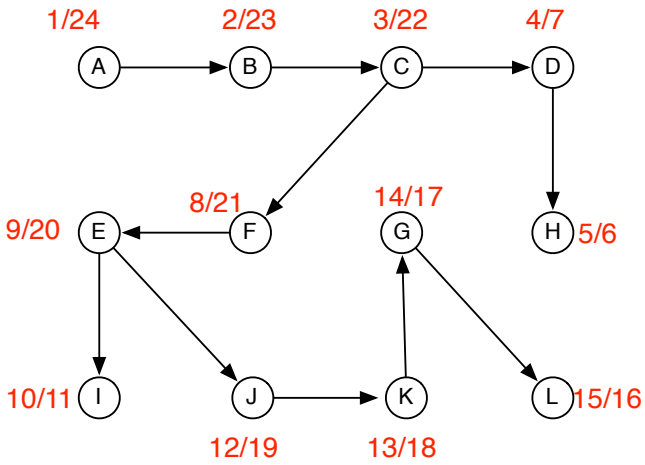
2.2 Betragt nedenstående graf G.



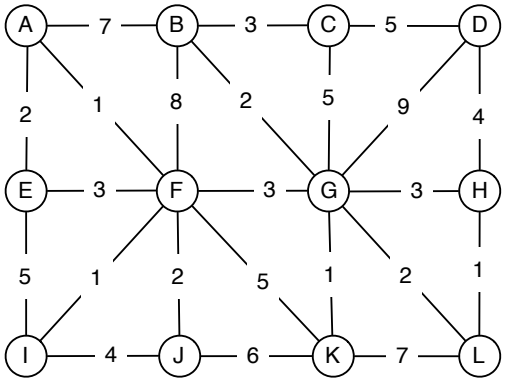
a) Angiv et BFS træ for grafen G når BFS gennemløbet starter i knuden A. Angiv BFS-dybde/lag for hver knude. Det antages at incidenslisterne er sorteret i alfabetisk orden.



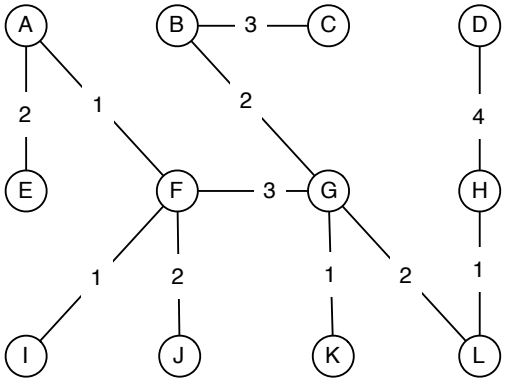
b) Angiv et DFS træ for grafen G, når DFS gennemløbet starter i knuden A. Angiv *discovery time* og *finishing time* for hver knude. Det antages at incidenslisterne er sorteret i alfabetisk orden.



2.3 Angiv et mindste udspændende træ i nedenstående graf.

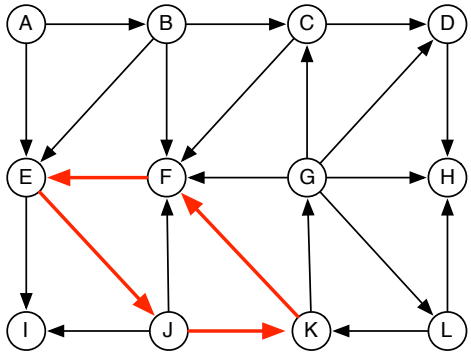


Angiv det mindste udspændende træ og værdien af det her:



Værdi: **22**

2.4 Er nedenstående graf en DAG (*directed acyclic graph*)? Hvis ja, så angiv en topologisk sortering af knuderne. Hvis nej, så forklar hvorfor.



Grafen er ikke en DAG, da den indeholder kredse (se kreds markeret med rødt).

Opgave 3 (modellering og anvendelse af algoritmer/datastrukturer)

Rejseselskabet ØerЯos planlægger rejser til øgruppen Algo. Nogle af øerne er forbundet med broer, andre må man sejle imellem. Der er X øer, B broer, og F færger. Hver færge sejler frem og tilbage mellem to øer.

Opgave 3.1: Rejser for søsyge

Da nogle af rejseselskabets kunder nemt bliver søsyge, ønsker firmaet at finde det største antal øer man kan besøge uden at skulle sejle imellem dem. På den måde kan de rejsende nøjes med at sejle til den første ø i gruppen og derefter komme til de andre øer via broerne.

Giv en effektiv algoritme der finder det største antal øer man kan besøge uden at sejle imellem dem.

Angiv køretiden af din algoritme i asymptotisk notation (ved hjælp af parametrene X, B og F) og argumenter for at den virker.

Svar Problemet kan modelleres som en uorienteret graf. Øer er knuder og broerne er kanter. Der er en knude for hver ø. Der er en kant mellem to knuder, hvis der er en bro mellem de to øer som knuderne svarer til. Der er X øer og B kanter.

Problemet svarer nu til at finde den største sammenhængskomponent i grafen. Dette kan gøres ved hjælp af enten BFS- eller DFS-algoritmen. Kør BFS (eller DFS) fra en knude. Modificer algoritmen så den har en tæller, der tælles op hver gang en ny knude mødes. Til sidst returneres antallet af mødte knuder, dvs. antallet af knuder i den sammenhængskomponent.

Hvis der stadig er knuder i grafen der ikke er besøgt, så kør BFS (eller DFS) igen fra en af disse (med tælleren nulstillet). Bliv ved med at køre BFS (DFS) indtil alle knuder er besøgt. Nu kendes størrelserne af alle sammenhængskomponenterne og det største tal returneres.

BFS-algoritmen tager $O(X+B)$ tid. Modifikationen ændrer ikke på køretiden, da tælleren kan opdateres i konstant tid i hver iteration (det samme gælder DFS).

Opgave 3.2: Billigste færgeture

Andre af rejseselskabets kunder ønsker at besøge en gruppe af Y øer, hvor der ikke er nogle broer imellem. Det er muligt at besøge alle øerne ved at tage færger imellem dem. Firmaet kender prisen for alle færgeture. Alle billetterne til færgerne er returbilletter. Dvs. man betaler for både at sejle frem og tilbage når man køber en færgebillet. Der er G færgeruter. Firmaet vil gerne tilbyde den billigste rejse, så de vil gerne minimere den pris man betaler for færgebilletter.

Giv en algoritme der finder ud af hvilke færgebilletter der skal købes for at alle Y øer kan besøges, og så den samlede pris for færgebilletter bliver mindst mulig. Angiv køretiden af din algoritme i asymptotisk notation (ved hjælp af parametrene Y og G) og argumenter for at den virker.

Svar Problemet modelleres som en vægtet uorienteret graf. Øer er knuder og færgertuerne er kanter. Der er en knude for hver ø. Der er en kant mellem to knuder, hvis der er en færgerute mellem de to øer som knuderne svarer til. Vægten af kanten er prisen for færgebilletten. Der er Y øer og G kanter. Da priserne er for en returbillet, svarer problemet til at finde det mindste udspændende træ i grafen. Når man har fundet et mindste udspændende træ kan man besøge alle øer ved at følge en rute der svarer til et trægennemløb af træet. Hver kant besøges kun 2 gange (en gang i hver retning) og det svarer til at der betales for netop en returbillet for hver færgerute i det mindste udspændende træ. Prisen for færgebilletter i alt bliver derfor vægten af det mindste udspændende træ. Denne kan findes i lineær tid ($\mathrm{O}(Y)$ tid) ved et trægennemløb når først MST er fundet.

Det er ikke muligt at gøre det billigere end prisen for et mindste udspændende træ, da enhver rute der besøger alle øerne må være sammenhængende, og et mindste udspændende træ er den billigste sammenhængende delgraf i grafen.

Hvis Prims algoritme bruges bliver køretiden $O(G \log Y)$.

Opgave 3.3: Billigste tur til alle øer

Som i forrige opgave ønsker firmaet at minimere den pris man betaler for færgebilletter. Men nu er der broer mellem nogle af øerne. Da broerne er gratis er der ingen grund til at tage en færge mellem to øer hvis der er en bro. Der er X øer, B broer, og F færger.

Giv en algoritme der finder ud af hvilke færgebilletter der skal købes for at alle X øer kan besøges, og så den samlede pris for færgebilletter bliver mindst mulig. Angiv køretiden af din algoritme i asymptotisk notation (ved hjælp af parametrene X, B og F) og argumenter for at den virker.

Svar Problemet modelleres som en graf G, hvor alle øer er knuder og kanterne svarer til broer og færgeruter. Kanterne der svarer til broer har vægt 0, og kanterne der svarer til færgeruter har vægt lig med prisen for billetten. Dvs. der er en kant mellem to knuder hvis der er en bro eller færgerute imellem dem.

Problemet kan nu løses ved hjælp af algoritmen fra opgave 3.2.

Grafen G har X knuder og højst $F + B$ kanter. Bruges Prims algoritme til at finde MST i G tager $O((F + B) \log X)$ tid.

Alternativt svar Problemet modelleres som en graf G_1, hvor alle øer er knuder og kanterne svarer til broer som i opgave 3.1. Alle sammenhængskomponenter findes som i opgave 3.1. Der laves nu en ny graf G_2, hvor knuderne er sammenhængskompenterne i G_1. Kanterne i G_2 er færgeruterne. Der er en kant mellem to knuder u og v hvis der er en færgerute mellem en ø i u og en ø i v. Vægten af kanten er prisen for den billigste sådanne færgerute. Problemet kan nu løses ved hjælp af algoritmen fra opgave 3.2.

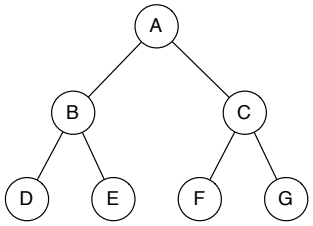
At finde alle sammenhængskomponenterne i G_1 tager $O(X + B)$ tid. Når sammenhængskomponenterne er fundet tager det $O(X + F)$ tid at konstruere G_2. Grafen G_2 har højst X knuder og højst F kanter. Bruges Prims algoritme til at finde MST i G_2 tager denne del $O(F \log X)$ tid.

I alt bliver køretiden: $O(X + B) + O(X + F) + O(F \log X) = O(X + B + F \log X)$.

Opgave 4 (Træer)

4.1 Trægennemløb

Betragt nedenstående træ.



Angiv hvilke af følgende sekvenser af bogstaver der bliver udskrevet ved et preorder, inorder og postorder gennemløb af ovenstående træ.

[1] A B C D E F G [2] G F E D C B A [3] A B D E C F G

[4] D B E A F C G [5] D E B F G C A [6] C B D A F E G

Preorder: **3**

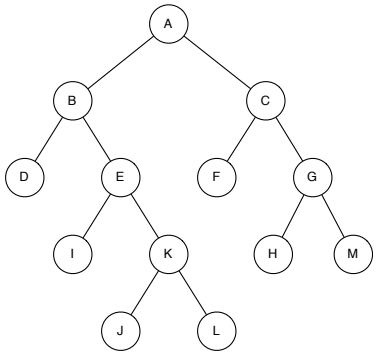
Inorder: **4**

Postorder: **5**

4.2 Korteste sti

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.

4.2.1 Betragt nedenstående træ.



Hvad er længden af den korteste rod-til-blad sti i træet: **2**

Hvad er længden af den længste rod-til-blad sti i træet: **4**

4.2.2 Betragt følgende algoritme:

Algorithm 3 TRÆ(x)

```
1: if (x = NIL or left[x] = NIL) then
2:     return 0
3: else
4:     return TRÆ(left[x]) + 1
5: end if
```

Forklar hvad algorimen TRÆ(x) beregner når den bliver kørt med rodknuden til et vægtet binært træ T som input?

Svar Den returnerer længden af stien der går fra roden og ned til bladet længst til venstre.

Lav algoritmen om så den i stedet returnerer længden af den korteste rod-til-blad sti i træet. Angiv køretiden af din algoritme i asymptotisk notation og argumenter for at den virker.

Svar

Algorithm 4 TRÆ(x)

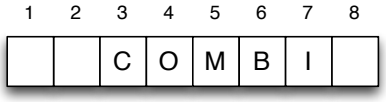
```
1: if (x = NIL or left[x] = NIL) then
2:     return 0
3: else
4:     return min{TRÆ(left[x]), TRÆ(right[x])}+1
5: end if
```

Algoritmen finder længden af den korteste rod-til-blad sti i børnenes deltræer, tager minimum af de to og lægger en til svarende til kanten fra knuden x til barnet. Hvis det er et tomt træ returneres 0. Hvis x er et blad returneres også 0.

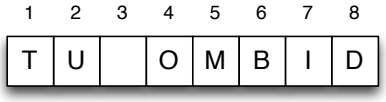
Køretiden af algoritmen er $\Theta(n)$ hvor n er antallet af knuder i træet. Hvert kald tager konstant tid og der kaldes kun en gang for hver knude.

Opgave 5 (datastrukturer)

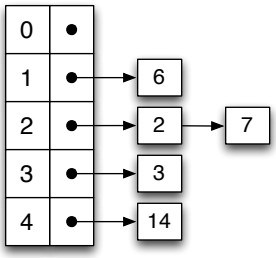
5.1 Betragt nedenstående kø K implementeret ved hjælp af en tabel (et array). K.head = 3 og K.tail = 8.



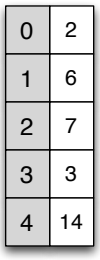
Angiv hvordan køen ser ud efter følgende operationer: Enqueue(D), Enqueue(T), Dequeue(), Enqueue(U).



5.2 Lad H være en hægtet hashtabel (chained hashing) af størrelse 5 med hashfunktion $h(x) = x \mod 5$. Angiv hvordan hashtabellen H ser ud efter indsættelse af tallene 6, 7, 3, 14, 2

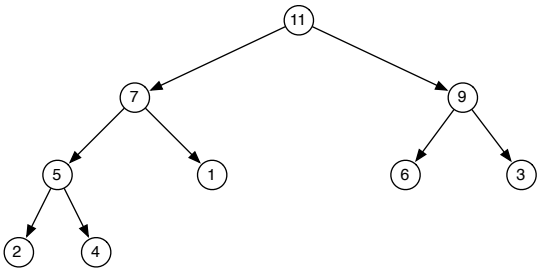


5.3 Lad H være en hashtabel med linær probering (linear probing) af størrelse 5 med hashfunktion $h(x) = x \mod 5$. Angiv hvordan hashtabellen H ser ud efter indsættelse af tallene 6, 7, 3, 14, 2

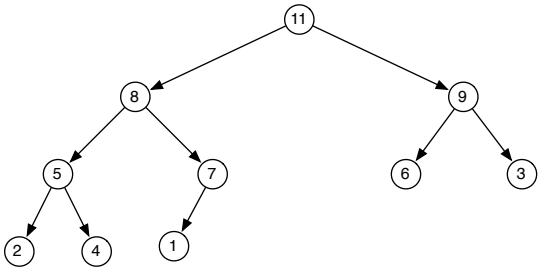


5.5 Denne opgave omhandler binære max-hobe.

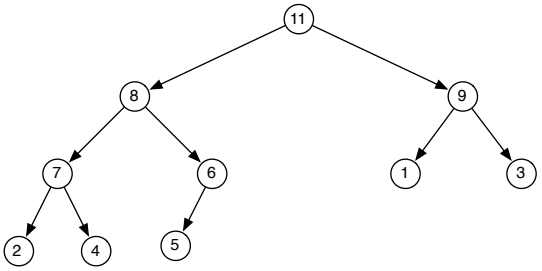
Opgave a Angiv hvordan den binære hob nedenfor ser ud efter indsættelse af et element med nøgle 8.



Svar



Opgave b Angiv hvordan den binære hob nedenfor ser ud efter én Extract-Max operation.



Svar