Danmarks Tekniske Universitet

Skriftlig prøve, den 26. maj 2009.

Kursusnavn Algoritmik og datastrukturer I Kursus nr. 02105.

Tilladte hjælpemidler: Alle skriftlige hjælpemidler.

Vægtning af opgaverne: Opgave 1 - 24%, Opgave 2 - 16%, Opgave 3 - 23%, Opgave 4 - 12 %, 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 benyttes ekstra papir som så vedlægges opgavebesvarelsen.

Opgave 1 (kompleksitet)

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

	Ja	*Nej*
$n^7 = O(n^3)$	☐	☒
$n(\log n)^3 = O(n^2)$	☒	☐
$2^n = O(n^2)$	☐	☒
$n^2 + \frac{1}{2}n = \Omega(n)$	☒	☐
$n(n-3)/17 = \Theta(n^2)$	☒	☐

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

$4\sqrt{n}$

$2/\log n$

$\frac{1}{2}n^3$

$\frac{2}{3}n$

$(\log n)^5$

Svar: $2/\log n,\ (\log n)^5,\ 4\sqrt{n},\ \frac{2}{3}n,\ \frac{1}{2}n^3$

1.3 Antag at du har en algoritme hvis køretid er præcist $5n^2$. Hvor meget langsommere kører algoritmen hvis du forbdobler inputstørrelsen?

☐ A dobbelt så langsom ☒ **B 4 gange langsommere** ☐ C 5 gange langsommere

☐ D 10 gange langsommere ☐ E 20 gange langsommere

1.4 Betragt nedenstående algoritme.

Algoritme Løkke1(n)
1. $x = 1$
2. **for** $i = 1$ **to** n
3. **for** $j = 1$ **to** n
4. **for** $k = 1$ **to** n
5. $x = x + 1$

a) Køretiden af algoritmen er

☐ A $\Theta(\log n)$ ☐ B $\Theta(n)$ ☐ C $\Theta(n \log n)$ ☐ D $\Theta(n^2 \log n)$ ☒ $\underline{\Theta(n^3)}$

☐ F $\Theta(n^{3/2})$ ☐ G $\Theta(2^n)$ ☐ H $\Theta(n^4)$ ☐ I $\Theta(\sqrt{n})$

b) Hvis linie 4 ændres til "**for** $k = j$ **to** n" så bliver køretiden:

[A] asymptotisk langsommere

[X] **asymptotisk den samme**

[C] asymptotisk hurtigere

1.5 Betragt nedenstående algoritme.

Algoritme Løkke2(n)
1. $i = 1$
2. **while** $i \leq n$ **do**
3. $j = 1$
4. **while** $j \leq n$ **do**
5. $j = j + 1$
6. $i = 2i$

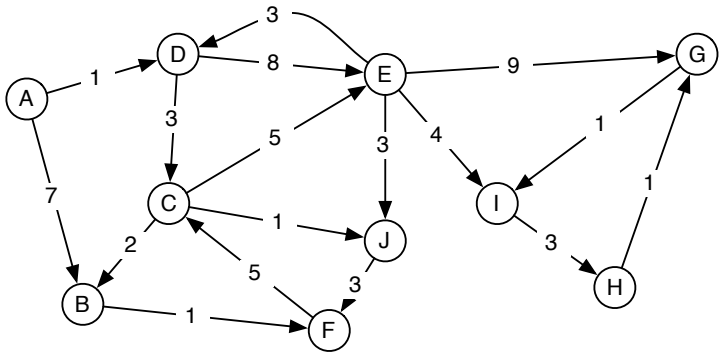
Køretiden af algoritmen er

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

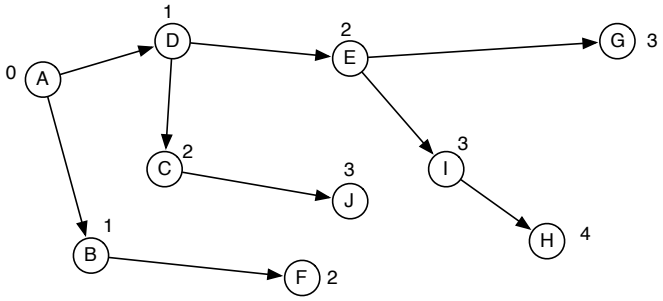
[F] $\Theta(n^{3/2})$ [G] $\Theta(2^n)$ [H] $\Theta(n^4)$ [I] $\Theta(\sqrt{n})$

Opgave 2 (grafer)

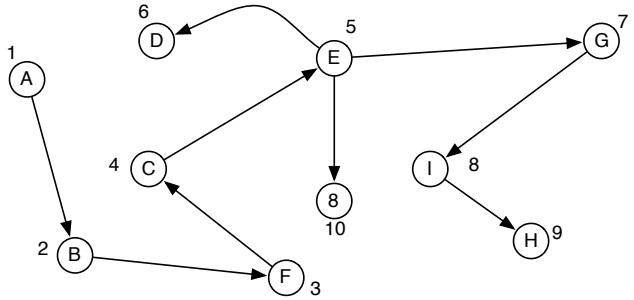
2.1 Betragt nedenstående graf G med ikke-negative kantvægte. Det antages at incidenslisterne er sorteret alfabetisk.



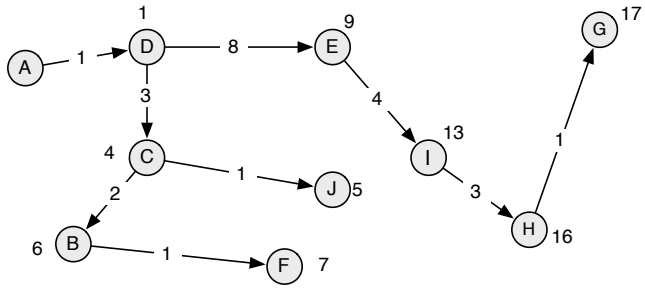
a) Angiv et BFS træ for grafen G når BFS genemløbet starter i knuden A. Angiv BFS-dybde/lag for hver knude. Det antages at incidenslisterne er sorteret i alfabetisk orden.



b) Angiv et DFS træ for grafen G, når DFS genemløbet starter i knuden A. Angiv en DFS nummerering af knuderne (en DFS nummerering er den rækkefølge knuder bliver besøgt i). Det antages at incidenslisterne er sorteret i alfabetisk orden.

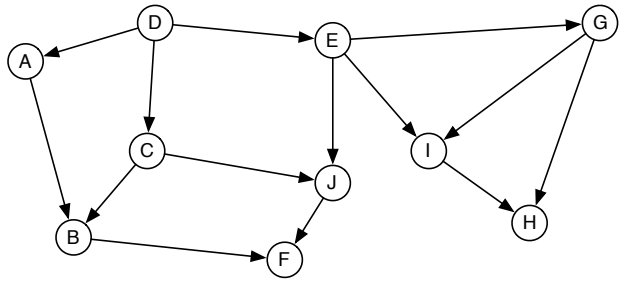


c) Angiv et korteste veje træ for grafen G når korteste veje beregningen sker med hensyn til startknuden A. Angiv for hver knude afstanden fra knuden A.



Obs. Det er også korrekt at have en kant fra C til E i stedet for D til E.

2.2 Betragt nedenstående DAG D.



Hvilke af følgende rækkefølger angiver en topologisk sortering af knuderne i grafen D:

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

[X] **D E A C B J F G I H** [5] D A E C B J G H [6] D E A C F B J G I H

Opgave 3 (modellering, anvendelse og analyse af algoritmer)

Du er konsulent for langtursbusfirmaet "Blåhundebusserne" der ønsker at lave en hjemmeside hvor folk kan finde og bestille deres rejse. Man skal både kunne søge på den billigste rejse mellem to valgfrie destinationer og på den rejse med færrest antal omstigninger (busskift). Der er S forskellige stoppesteder/stationer og B forskellige busruter. For hver busrute kender du startsted og destination, samt prisen for at benytte ruten. En bus stopper ikke undervejs, men kører direkte fra startsted til destination. Du skal ikke tage hensyn til tidspunkt for afgang og ankomst i denne opgave.

3.1 Giv en effektiv algoritme der finder den billligste rejse mellem to givne destinationer s og t. Rejsen kan være en kombination af flere forskellige busruter. Prisen for rejsen bliver så summen af priserne for de ruter der benyttes. Angiv køretiden af din algoritme i asymptotisk notation. Din analyse skal være så tæt som mulig.

Problemet kan modelleres med en vægtet graf. Stationerne er knuder og ruterne er kanter. Der er en kant fra knude A til knude B, hvis der er en busrute fra station A til station B. Kantvægten på en kant er prisen for den tilsvarende busrute.

Problem kan nu løses vha. en korteste vej forespørgsel: Den billigste rejse fra A til B svarer til den korteste vej fra A til B i den vægtede graf. Vi kan bruge Dijkstras algoritme til korteste vej.

Lad m være antal busruter (= antal kanter) og $n$være antal stationer (= antal knuder). Det tager linær tid at lave grafen, dvs. $O(m+n)$: Vi laver en graf med incidenslister. For at lave knudetabellen gives hver station et nummer (der kan bruges en hashtabel). Incidenslisterne laves nu ved at gennemgå hver busrute og indsætte en kan ml. startsted og destination med vægt lig prisen for ruten. Dijkstras algoritme tager $O(n+m \log n)$ tid. I alt tager det $O(n+m \log n)$ tid.

3.2 Giv en effektiv algoritme der finder den rejse mellem to givne destinationer s og t der har færrest antal omstigninger. Angiv køretiden af din algoritme i asymptotisk notation. Din analyse skal være så tæt som mulig.

Problemet kan modelleres med samme graf som ovenfor. Den rejse med færrest omstigninger mellem A og B svarer til den sti mellem A og B med færrest kanter. Vi kan derfor løse problemet vha. BFS-agoritmen. Som ovenfor tager det $O(m+n)$ tid at konstrurere grafen og BFS tager også $O(m+n)$ tid, da vi bruger incidenslister til at repræsentere grafen.

3.3 Firmaet ønsker også hjælp til at finde ud af hvornår busserne skal stoppe og tanke. Givet en busrute, kender vi placeringen af alle tankstationerne på ruten. Vi ved også hvor mange km k bussen kan køre på en fuld tank. Vi ønsker at lave så få stop for at tanke som muligt. Vi antager at bussen starter med en fuld tank. Vi kalder dette for *kør-og-tank* problemet.

Eksempel. Ruten er 300km. Bussen kan køre 110 km på en fuld tank. Tankstationerne på ruten ligger efter 10, 50, 80, 140, 180, 250, og 270 km.
I dette tilfælde skal bussen skal tanke mindst 3 gange, f.eks. efter 50, 140 og 250 km.

En af chaufførerne kommer med følgende strategi:

Grådig strategi: Kør så langt som muligt før der tankes.

a) Angiv hvor bussen stopper for at tanke hvis strategien bruges på eksemplet ovenfor.

Efter 80, 180 og 270 km.

b) Giv et argument for at den grådige strategi er korrekt eller giv et modeksempel.

Den grådige strategi er optimal. Lad L være længden af ruten. Vi siger at en sekvens af tankstationer er lovlig, hvis afstanden mellem to efter hinanden følgende stationer er højst k, og afstanden fra startsted til første tankstation, samt afstangen fra sidste tankstation til destinationen er højst k.

Lad $S = \{x_{s_1}, \ldots, x_{s_n}\}$ være sekvensen af tankstationer valgt af den grådige strategi. Lad $R = \{x_{t_1}, \ldots, x_{t_m}\}$ være en anden lovlig sekvens. Vi vil vise at $n \leq m$. Da dette gælder for enhver anden sekvens medfører det at S er en optimal sekvens. Vi vil først vise at efer hvert stop er bussen i S mindst lige så som i løsning R:

(*) For $j = 1, 2, \ldots, m$ har vi $x_{s_j} \geq x_{t_j}$.

Vi bruger induktion over j til at vise (*).
Basistilfælde $j = 1$: følger direkte af definitionen af den grådige strategi.
Induktionsskridt: Antag at det gælder for $i < j$ (induktionshypotesen). Vi har

$$\begin{aligned} k &\geq x_{t_j} - x_{t_{j-1}} && \text{da } R \text{ er lovlig} \\ &\geq x_{t_j} - x_{s_{j-1}} && \text{da } x_{s_{j-1}} \geq x_{t_{j-1}} \text{ i følge induktionshypotesen} \end{aligned}$$

Altså kan bussen nå fra $x_{s_{j-1}}$til x_{t_j}på en fuld tank. Da bussen i følge den grådige strategi kører så langt som muligt før den tanker har vi at $x_{s_j} \geq x_{t_j}$.

(*) medfører at $x_{s_m} \geq x_{t_m}$. Antag at $m \leq n$. Så er $x_{s_m} \leq L - k$, ellers ville bussen ikke behøve at tanke på tankstation $x_{s_{m+1}}$. Men så er $x_{t_m} \leq x_{s_m} \leq L - k$, og så er R ikke lovlig. Altså må $m \geq n$ og den grådige strategi er optimal.

Opgave 4 (rekursion)

4.1 Denne opgave omhandler en algoritme for fakultetsfunktionen ($n! = n \cdot (n-1) \cdots 1$). Nedenfor er 3 forsøg på at lave en rekursiv algoritme der beregner $n!$.

Algoritme Fak1(n)
  if $n = 1$
    return n
  $x = n \cdot (\text{Fak1}(n) - 1)$
  return x

Algoritme Fak2(n)
  if $n = 0$
    return n
  $x = n \cdot \text{Fak2}(n-1)$
  return x

Algoritme Fak3(n)
  if $n = 1$
    return n
  return $n \cdot \text{Fak3}(n-1)$

Hvilken af algoritmerne beregner $n!$ korrekt når n er et positivt heltal: Fak3

4.2 Nedenstående algoritme tager et positivt heltal som input.

Algoritme Rekur(n)
1. **if** $n \leq 0$
2. **return** 0
3. **return** n + Rekur($n-1$) + 2

Giv iterativ variant af algoritmen:

Rekur-Iter(n)
1. $x = 0$, $i = 1$.
2. **while** $i \leq n$ **do**
3. $x = x + 2 + i$
4. $i = i + 1$
5. **return** x

Opgave 5 (datastrukturer)

5.1 I hvilke af nedenstående datastrukturer kan man lave søgning i worst case $O(\log n)$ tid, hvor n er antal elementer i datastrukturen (du må gerne sætte mere end et kryds)?

[1] Hashtabel	[2] sorteret hægtet liste	[3] hob
[X] **sorteret tabel**	[5] usorteret tabel	[6] binært søgetræ

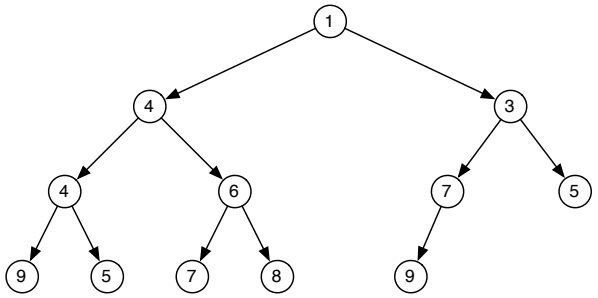
Jeg har givet 3 point hvis man har sat kryds ved både sorteret tabel og binært søgetræ, og 4 point hvis man kun har sat kryds ved sorteret tabel, da opgaven var utilsigtet svær.

5.2 I hvilke af nedenstående datastrukturer kan man lave indsættelse i worst case $O(1)$ tid, hvor n er antal elementer i datastrukturen (du må gerne sætte mere end et kryds)?

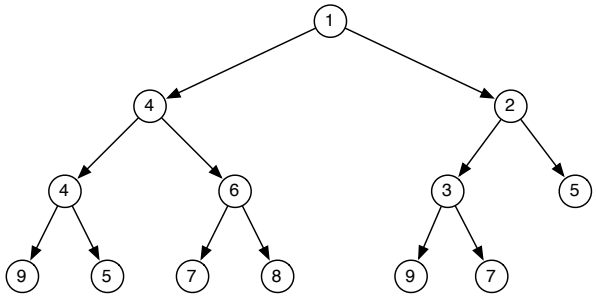
[X] **Hashtabel**	[X] **usorteret hægtet liste**	[3] hob
[4] sorteret tabel	[5] binært søgetræ	

5.3 Denne opgave omhandler hobe, som beskrevet i bogen i afsnit 2.5.

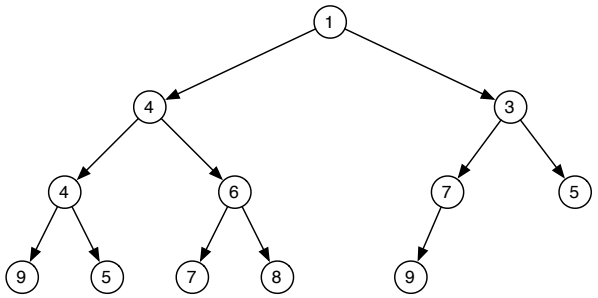
a) Angiv hvordan hoben nedenfor ser ud efter indsættelse af et element med nøgle 2.



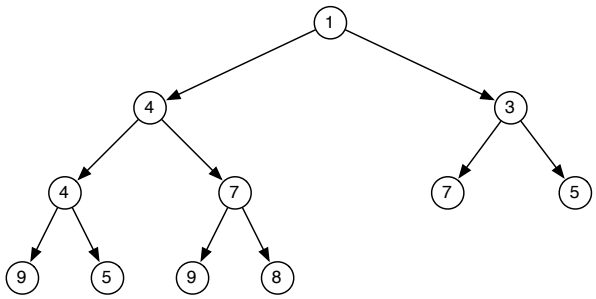
Løsning:



b) Angiv hvordan hoben nedenfor ser ud efter sletning af elementet med nøglen 6.



Løsning:



c) Giv en algoritme der i lineær tid checker om nøglerne i knuderne i et komplet binært træ opfylder hoborden (se side 59 i bogen for definitionen af hoborden).

Algoritmen tager en knude som input og returnerer 1 hvis træet rodfæstet i v opfylder hoborden og 0 ellers.

Agoritme Hoborden(v)

```
if v = NIL { return 1 }
else {
  if (left(v) ≠ NIL and key[left(v)] < key[v])
    { return 0 }
  if (right(v) ≠ NIL and key[right(v)] < key[v])
    { return 0 }
  return min(Hoborden(left(v)), Hoborden(right(v)))
}
```

Et kald til algoritmen tager konstant tid i sig selv (uden de rekursive kald). Det tager konstant tid at checke hver betingelse i if-sætningerne og at returnere 0 eller 1. For hver knude laves der to rekursive kald. Men for enhver knude u kaldes Hoborden(u) kun en gang. Algoritmen tager derfor lineær tid i antallet af knuder i træet.

5.4 I kør-og-tank problemet fra opgave 3.3 kender vi placeringen af tankstationerne på ruten. Antag at placeringen af en tankstation t er angivet som afstanden d_t fra starten til t. Givet en bus' position p og hvor mange km k den kan køre på en fuld tank ønsker vi at finde den tankstation der ligger længst væk af de tankstationer bussen kan nå inden den løber tør for diesel. Dvs. givet p og k ønsker vi at finde t således $d_t \leq p + k$ og d_t er størst mulig. Lad n være antallet af tankstationer. Hvilken datastruktur kan bruges til at finde det ønskede t i $O(\log n)$ tid givet p og k?

Der kan bruges et binært søgetræ eller en sorteret tabel. Forespørgslen find det maksimale $t$således at $d_t \leq p + k$ svarer til en predecessor forespørgsel: pred($p + k$) i en datastruktur der indeholder alle tankstationerne som elementer med placeringen af tankstationerne som nøgler. En predecessor forespørgsel kan besvares i $O(\log n)$tid i et binært søgetræ. I en sorteret tabel kan den besvares i $O(\log n)$ tid ved binær søgning.