EKSAMEN

Algoritmer og Datastrukturer

Fredag 22. januar 2021, 9:00–11:00

Institut for Datalogi, Naturvidenskabelige Fakultet, Aarhus Universitet

Antal sider i opgavesættet (incl. forsiden): 14

Tilladte hjælpemidler:
Alle, inklusive internet.
Det er ikke tilladt at kommunikere med andre under eksamen.

Studienummer :

Navn :

Dokumentet skal udfyldes med Adobe Acrobat,
som kan hentes fra `https://get.adobe.com/reader/`

Vejledning og pointgivning

Dette eksamenssæt består af en mængde multiple-choice-opgaver.

Opgaverne besvares på opgaveformuleringen **som afleveres**.

For hver opgave er angivet opgavens andel af det samlede eksamenssæt.

Hvert delspørgsmål har præcist et rigtigt svar.

For hvert delspørgsmål må du vælge **max ét svar** ved at afkrydse den tilsvarende rubrik.

Et delspørgsmål bedømmes som følgende:

- Hvis du sætter kryds ved det rigtige svar, får du 1 point.
- Hvis du ikke sætter nogen krydser, får du 0 point.
- Hvis du sætter kryds ved et forkert svar, får du $-\frac{1}{k-1}$ point, hvor k er antal svarmuligheder.

For en opgave med vægt $v\,\%$ og med n delspørgsmål, hvor du opnår samlet s point, beregnes din besvarelse af opgaven som:

$$\frac{s}{n} \cdot v\,\%$$

Bemærk at det er muligt at få negative point for en opgave.

Opgave 1 (Asymptotisk notation, 6 %) check

I det følgende angiver $\log n$ 2-tals-logaritmen af n.

	Ja	Nej
$3n^{3/2} + \log n$ er $O(n^{2/3})$		
$2(\log n)^4$ er $O(n^2)$		
$\sqrt{n} \cdot \log n$ er $O(n)$		
$6n^{3/2}$ er $O(8^{\log n})$		
$4\log(n^6)$ er $O((\log n)^2)$		
$2^{2\log n}$ er $O(\log(n!))$		
n^2 er $O(n^{3/2})$		
$n^{0.1}$ er $O(n)$		
n er $O(n^{1/3})$		
$\sqrt{n}$ er $\Theta(n \cdot \log n)$		
$n^{2/3}$ er $\Omega(n)$		
n^2 er $\Theta(n^{0.1})$		

Opgave 2 (Analyse af løkker, 6 %) check

```
Algoritme loop1(n)
  s = 1
  for i = 1 to n
    for j = 1 to i
      for k = j to i
        s = s + 1
```

```
Algoritme loop2(n)
  i = 0
  j = n
  while i < j
    i = i + 2
    j = j + 1
```

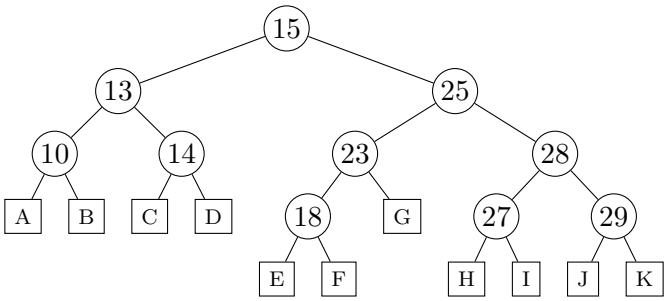
```
Algoritme loop3(n)
  i = 1
  j = n
  while i < j
    i = 2 * i
    j = j + n
```

```
Algoritme loop4(n)
  i = n
  j = 0
  while i > 0
    if j < i
      j = j + 1
    else
      j = 0
      i = i - 1
```

Angiv for hver af ovenstående algoritmer udførselstiden som funktion af n i Θ-notation.

	$\Theta(n^3)$	$\Theta(n \log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta((\log n)^2)$	$\Theta(n^2)$	$\Theta(2^n)$	$\Theta(\log n)$
loop1								
loop2								
loop3								
loop4								

Opgave 3 (Indsættelser i søgetræer, 4 %) **check**



Angiv i hvilke blade A–K i ovenstående ubalancerede binære søgetræ elementerne 22, 17, 26, 30 og 16 skal indsættes (det antages at før hver indsættelse indeholder træet kun ovenstående ti elementer).

	A	B	C	D	E	F	G	H	I	J	K
INSERT(22)											
INSERT(17)											
INSERT(26)											
INSERT(30)											
INSERT(16)											

Opgave 4 (Build-Max-Heap, 4 %) **check**

1	2	3	4	5	6	7	8	9
6	1	3	8	2	9	5	7	4

Hvad er resultat af BUILD-MAX-HEAP på ovenstående array ?

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
6	8	9	7	2	3	5	1	4

1	2	3	4	5	6	7	8	9
9	8	6	7	2	3	5	1	4

1	2	3	4	5	6	7	8	9
9	7	8	6	2	3	5	1	4

Opgave 5 (Heap-Extract-Max, 4 %) **check**

1	2	3	4	5	6	7	8	9	10	11	12	13
22	19	17	14	16	15	13	3	9	2	8	4	11

Hvad er resultat af HEAP-EXTRACT-MAX på ovenstående max-heap ?

1	2	3	4	5	6	7	8	9	10	11	12	13
19	16	17	14	8	15	13	3	9	2		4	11

1	2	3	4	5	6	7	8	9	10	11	12
19	16	17	14	8	15	13	3	9	2	4	11

1	2	3	4	5	6	7	8	9	10	11	12
19	16	17	14	8	15	13	3	9	2	11	4

1	2	3	4	5	6	7	8	9	10	11	12
19	16	17	14	11	15	13	3	9	2	8	4

1	2	3	4	5	6	7	8	9	10	11	12
19	17	14	16	15	13	3	9	2	8	4	11

Opgave 6 (Lineær probing, 4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
	15				9	13	2			18

I ovenstående hashtabel af størrelse 11 er anvendt *linear probing* med hashfunktionen $h(k) = 3k \bmod 11$.

Angiv positionerne de fem elementer 4, 5, 6, 7 og 11 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 2, 9, 13, 15 og 18).

	0	1	2	3	4	5	6	7	8	9	10
INSERT(4)											
INSERT(5)											
INSERT(6)											
INSERT(7)											
INSERT(11)											

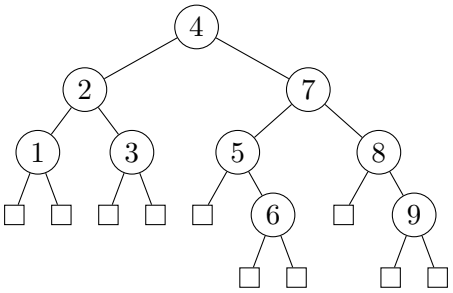
Opgave 7 (Merge-Sort, 4 %) **check**

Antag MERGE-SORT udføres på et input af størrelse n og indeholdende to elementer x og y. Hvad er worst-case antal sammenligninger af x med y under udførelsen af MERGE-SORT?

$\Theta(1)$ $\Theta(\log n)$ $\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$

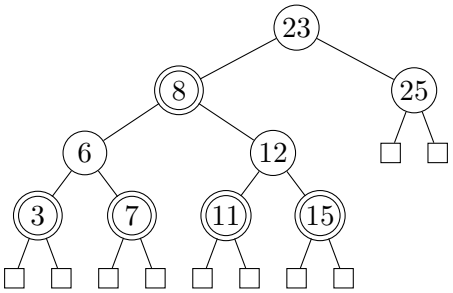
Opgave 8 (Rød-sort træ, 4 %) **check**

For hver af nedenstående delmængder, angiv om nedenstående binære træ er et lovligt rød-sort træ hvis netop disse knuder farves røde.

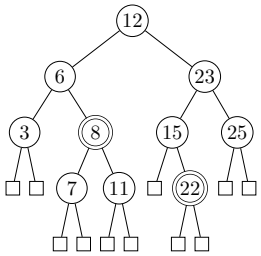


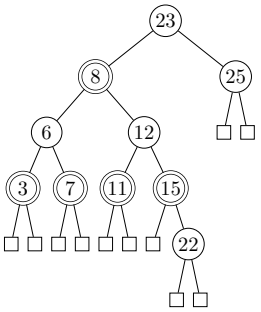
	Ja	Nej
6, 9		
2, 6, 7, 9		
2, 5, 6, 8, 9		
1, 3, 6, 7, 9		
4, 6, 9		

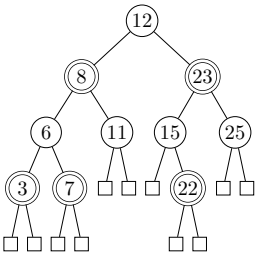
Opgave 9 (Indsættelse i rød-sort træer, 4 %) **check**

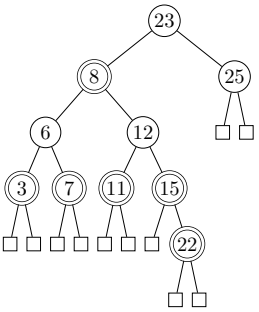


Angiv det resulterende rød-sorte træ når man indsætter 22 i ovenstående rød-sorte træ (dobbeltcirkler angiver røde knuder).





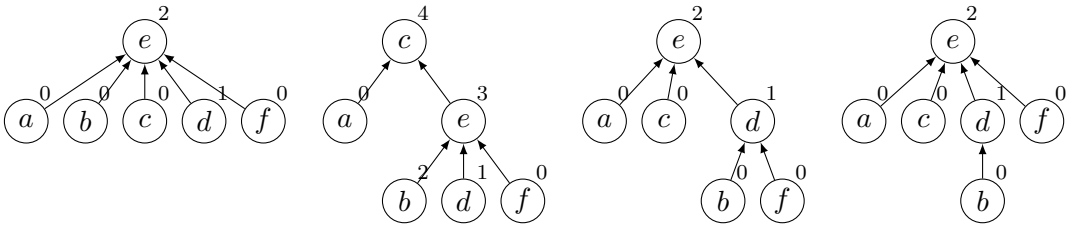




Opgave 10 (Union-find, 4 %) **check**

Angiv den resulterende union-find struktur efter nedenstående sekvens af operationer, når der anvendes union-by-rank og stikomprimering.

MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(f, d)
UNION(f, b)
UNION(a, e)
UNION(b, a)
UNION(f, c)
FIND-SET(a)



Opgave 11 (Rekursionsligninger, 4 %) **check**

Angiv løsningen for hver af nedenstående rekursionsligninger, hvor $T(n) = 1$ for $n \leq 1$.

$\Theta(\log n)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$ $\Theta(n^2 \log n)$ $\Theta(n^3)$

$T(n) = 4 \cdot T(n/4) + n$

$T(n) = 4 \cdot T(n/2) + 2$

$T(n) = 3 \cdot T(n/9) + 1$

$T(n) = T(n-1) + 3$

$T(n) = T(n/3) + 2$

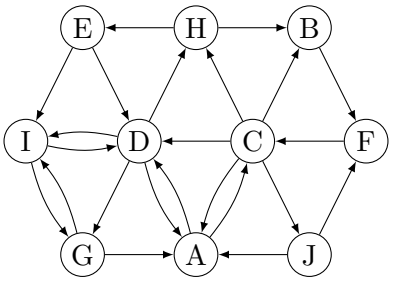
Opgave 12 (Alle-par-korteste-veje, 4 %) **check**

Antag vi har en orienteret graf med n knuder og postitivt vægtede kanter, hvor vi løbende tilføjer yderligere kanter. Vi ønsker at vedligeholde en afstands-tabel over de korteste afstande mellem alle par af knuder.

Hvad er den bedste worst-case tid man kan opnå for at opdatere afstands-tabellen, når man tilføjer en ny kant med positiv vægt til grafen?

$\Theta(1)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n \cdot \log n)$ $\Theta(n \cdot \sqrt{n})$ $\Theta(n^2)$ $\Theta(n^3)$

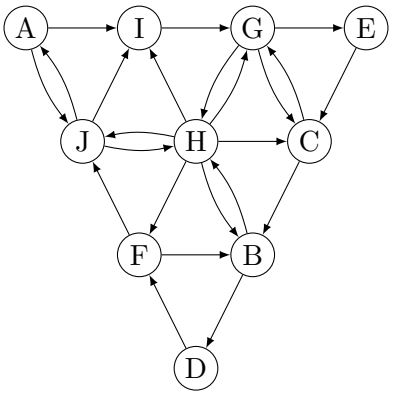
Opgave 13 (BFS, 4 %) check



For et bredde først gennemløb (BFS) af ovenstående graf **startende i knuden A**, angiv rækkefølgen knuderne bliver indsat i køen Q i BFS-algoritmen. Det antages, at grafen er givet ved incidenslister, hvor incidenslisterne er sorteret alfabetisk.

A C D B H J I G F E A C D B H J G I F E A C B F D G I H E J A C D J H B I G F E

Opgave 14 (DFS, 4 %) check



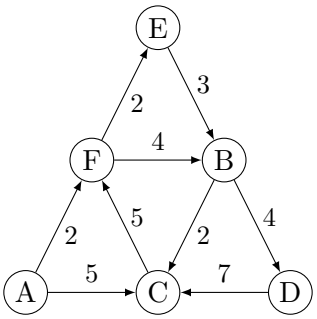
Betragt et dybde først gennemløb (DFS) af ovenstående graf, hvor DFS-gennemløbet starter i **knuden A**, hvor de udgående kanter til en knude besøges i alfabetisk rækkefølge. Angiv i hvilken rækkefølge knuderne får tildelt **finishing time**.

D F B E C H G J I A E H J F D B C G I A E G F D B C I H J A H J F D B C E G I A

Angiv for hver af nedenstående kanter hvilken type kanten bliver i DFS gennemløbet.

	Tree edge	Back edge	Cross edge	Forward edge
(E, C)				
(A, J)				
(J, A)				
(J, H)				

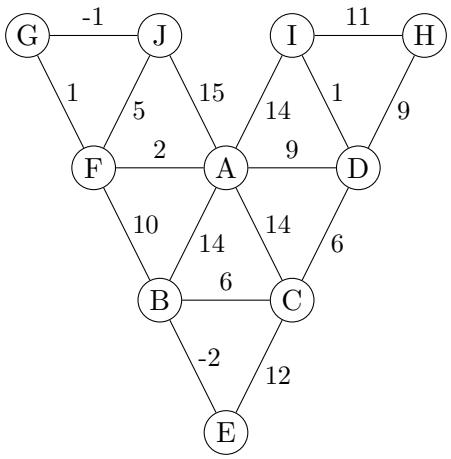
Opgave 15 (Dijkstras algoritme, 4 %) **check**



Antag Dijkstras algoritme anvendes til at finde korteste afstande fra **knuden A** til alle knuder i ovenstående graf. Angiv hvilken rækkefølge knuderne bliver taget ud af prioritetskøen i Dijkstra's algoritme.

A C F B E D A F E B C D A F E C B D A C F B D E

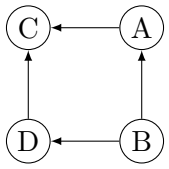
Opgave 16 (Prims algoritme, 4 %) **check**



Antag Prims algoritme anvendes til at finde et minimum udspændende træ for ovenstående graf, og algoritmen starter i **knuden A**. Angiv hvilken rækkefølge knuderne bliver inkluderet i det minimum udspændende træ (taget ud af prioritetskøen i Prims algoritme).

A F G J D I C B E H A F G J D I C H B E A F G J B E C D I H A F G J D I B E C H

Opgave 17 (Topologisk sortering, 4 %) **check**



Angiv for hver af nedenstående ordninger af knuderne i ovenstående graf om det er en lovlig topologisk sortering.

	Ja	Nej
D A B C		
B D A C		
A B D C		
C A D B		
B A D C		

Opgave 18 (Amortiseret analyse, 4 %) **check**

En binær max-heap understøtter Max-Heap-Insert og Heap-Extract-Max i worst-case tid $O(\log n)$, hvor n er antal elementer i heapen. Vi vil nu understøtte operationen Delete, der givet en pointer til et element i heapen, sletter elementet fra heapen. Vi implementerer Delete ved blot at markere elementet som slettet i worst-case $O(1)$ tid. Når vi udfører Heap-Extract-Max gentager vi denne indtil det første ikke-markerede element bliver returneret. Dvs. hvis Heap-Extract-Max sletter D markerede elementer bliver worst-case tiden $O((D+1)\log N)$, hvor N er antallet af markerede og ikke-markerede elementer i heapen.

Angiv for hver af nedenstående funktioner om de er en potentialefunktion, hvormed man kan argumentere for at operationerne Max-Heap-Insert, Delete, og Heap-Extract-Max tager amortiseret $O(\log N)$ tid, hvor M betegner antallet af markerede elementer i heapen.

	Ja	Nej
N		
M		
$N \cdot \log N$		
$M \cdot \log N$		
$N \cdot \log M$		
$M \cdot \log M$		

Opgave 19 (Invarianter, 4 %) **check**

Givet et positivt heltal n, så beregner nedenstående algoritme n^3.

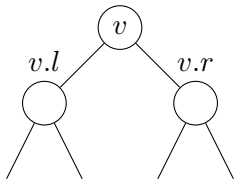
Algoritme Power3(n)
Inputbetingelse : Heltal $n \geq 1$
Outputkrav : $r = n^3$
Metode : $i \leftarrow 1$
$s \leftarrow 1$
$r \leftarrow 1$
$\{I\}$ **while** $i < n$ **do**
$i \leftarrow i + 1$
$s \leftarrow s + 2i - 1$
$r \leftarrow r + 3s - 3i + 1$

For hvert af følgende udsagn, angiv om de er en invariant I for algoritmen Power3.

	Ja	Nej
$1 \leq i < n$		
$1 \leq i \leq s \leq r$		
$r = i^3$		
$r = n^3$		
$s = s + 2i - 1$		

Opgave 20 (Udvidede søgetræer, 4 %) **check**

Betragt et søgetræ hvor hver knude v gemmer et tal $v.x$, og knuderne er ordnet venstre-mod-højre efter stigende $v.x$. Derudover gemmes i en knude v tre værdier $v.min$, $v.max$, og $v.closest$. Værdierne $v.min$ og $v.max$ er henholdsvis det mindst og største tal i v's undertræ, og $v.closest$ den mindste difference mellem to tal i v's undertræ. Hvis v's undertræ kun indeholder et tal er $v.closest = +\infty$.



Angiv hvorledes $v.closest$ og kan beregnes når min, max og $closest$ værdierne er kendt ved de to børn $v.l$ og $v.r$ (det kan antages at disse begge eksisterer).

$v.closest = \min(v.l.closest, v.r.x - v.l.x, v.r.closest)$

$v.closest = \min(v.l.closest, v.x - v.l.min, v.r.max - v.x, v.r.closest)$

$v.closest = \min(v.l.closest, v.x - v.l.max, v.r.min - v.x, v.r.closest)$

$v.closest = \min(v.l.closest, v.r.min - v.l.max, v.r.closest)$

$v.closest = v.r.closest - v.l.closest$

Dynamisk programmering

De næste fire opgaver vedrører at løse *strengkonkatenerings* problemet ved hjælp af dynamisk programmering.

Lad T være en streng af længde n, og lad $S_1, S_2, \ldots, S_k$ være k strenge. Vi ønsker at afgøre om T kan skrives som en konkatenation af strenge fra $S_1, S_2, \ldots, S_k$, hvor hvert S_i kan forekomme et vilkårligt antal gange (0, 1, eller flere gange). F.eks. kan strengen

$$T = \texttt{ABBBBDABABB}$$

skrives som en konkatenation af strenge fra

$$S_1 = \texttt{ABB} \qquad S_2 = \texttt{ACAA} \qquad S_3 = \texttt{BB} \qquad S_4 = \texttt{ABA} \qquad S_5 = \texttt{D}$$

For $1 \leq j \leq n+1$ lader vi $C(j)$ angive om $T[j\,..\,n]$ kan skrives som en konkatenation af strenge fra $S_1, \ldots, S_k$.

$C(j)$ kan bestemmes ved følgende rekursionsformel.

$$C(j) = \begin{cases} \textbf{sand} & \text{hvis } j = n+1 \\ \textbf{sand} & \text{hvis } j \leq n \text{ og der findes } i \text{ hvor} \\ & \qquad j + |S_i| - 1 \leq n \ \wedge\ C(j + |S_i|) \ \wedge\ S_i = T[j\,..\,j + |S_i| - 1] \\ \textbf{falsk} & \text{ellers} \end{cases}$$

De følgende 4 opgaver består i at udfylde 4 blokke i følgende algoritmeskabelon.

Algoritme Concatenation($T, \{S_1, \ldots, S_k\}$)
    $n = |T|$
    Opret tom tabel $D[1\,..\,n+1]$
    for ...
        ≪ **Opgave 21**: iterer over D ≫
        ≪ **Opgave 22**: beregn $D[j] = C(j)$ ≫
    ≪ **Opgave 23**: sæt *solution* til True eller False ≫
    if *solution* **then**
        ≪ **Opgave 24**: Udskriv en løsning ≫
    else
        print "Not possible"

Opgave 21 (4 %) **check**

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

Ja Nej

for $j = 1$ **to** $n + 1$

---

for $j = n + 1$ **to** 1 **step** -1

Opgave 22 (4 %) **check**

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

Ja Nej

if $j = n + 1$ **then**
  $D[j] = \text{True}$
else if $j + |S_i| - 1 \leq n$ **and** $D[j + |S_i|]$ **and** $T[j \,..\, j + |S_i| - 1] = S_i$ **then**
  $D[j] = \text{True}$
else
  $D[j] = \text{False}$

---

if $j = n + 1$ **then**
  $D[j] = \text{True}$
else
  $D[j] = \text{False}$
  for $i = 1$ **to** k
    if $j + |S_i| - 1 \leq n$ **and** $D[j + |S_i|]$ **and** $T[j \,..\, j + |S_i| - 1] = S_i$ **then**
      $D[j] = \text{True}$

$D[j] = \text{True}$
for $i = 1$ **to** k
  if $j + |S_i| - 1 \leq n$ **and** $D[j + |S_i|]$ **and** $T[j \,..\, j + |S_i| - 1] \neq S_i$ **then**
    $D[j] = \text{False}$

Opgave 23 (4 %) **check**

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

Ja Nej

$solution = D[n+1]$

$solution = D[1]$

$solution$ = True
for $j = 1$ **to** n
  if $D[j]$ = False **then**
    $solution$ = False

Opgave 24 (4 %) **check**

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

Ja Nej

$j = 1$
while $j \leq n$ **do**
  for $i = 1$ **to** k
    if $D[j + |S_i|]$ **then**
      $t = i$
  print S_t
  $j = j + |S_t|$

$j = n + 1$
while $j > 1$ **do**
  for $i = 1$ **to** k
    if $j - |S_i| + 1 \geq 1$ **and** $D[j - |S_i|]$ **and** $T[j - |S_i| + 1 \,..\, j] = S_i$ **then**
      $t = i$
  print S_t
  $j = j - |S_t|$

$j = 1$
while $j \leq n$ **do**
  for $i = 1$ **to** k
    if $j + |S_i| - 1 \leq n$ **and** $D[j + |S_i|]$ **and** $T[j \,..\, j + |S_i| - 1] = S_i$ **then**
      $t = i$
  print S_t
  $j = j + |S_t|$