EKSAMEN

Algoritmer og Datastrukturer

Mandag 8. januar 2024, 9:00–11:00

Institut for Datalogi, Naturvidenskabelige Fakultet, Aarhus Universitet

Antal sider i opgavesættet (incl. forsiden): 14

Tilladte medbragte hjælpemidler: **Ingen**

Kun det separate svarark skal udfyldes og afleveres.

Denne opgaveformulering skal *ikke* afleveres.

Vejledning og pointgivning

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

Svarene på opgaverne angives på det **separate svarark 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 notationer, 6 %)

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

	Ja	Nej	
$\sqrt{n}$ er $O(\sqrt{n} \cdot \log n)$	A	B	1.1
$4n + (\log n)^2$ er $O((\log n)^6)$	A	B	1.2
$n^{0.01}$ er $O(4)$	A	B	1.3
$(\log n)^3$ er $O(8^{\log n})$	A	B	1.4
$n!$ er $O(n^2)$	A	B	1.5
$8^{\log n}$ er $O(8 \cdot \log n)$	A	B	1.6
$\log n$ er $O(\sqrt{n} \cdot \log n)$	A	B	1.7
$\sqrt{n} + n^2 \cdot \log n$ er $O(n^3)$	A	B	1.8
$n^{\sqrt{n}}$ er $O(n!)$	A	B	1.9
$n^{2/3}$ er $\Omega(n \cdot \log n)$	A	B	1.10
$(\log n)^2$ er $\Omega(n)$	A	B	1.11
$n \log n$ er $\Theta(n^2)$	A	B	1.12

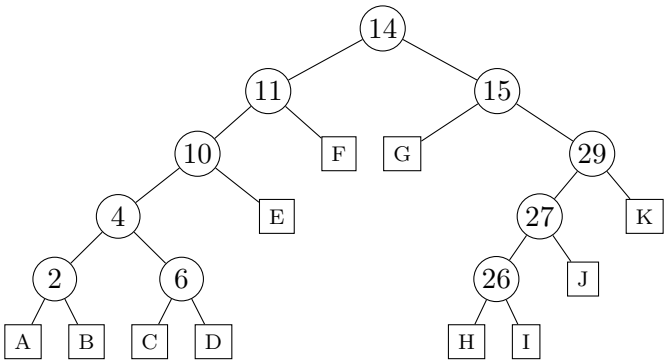
Opgave 2 (Analyse af løkker, 6 %)

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

Algoritme loop2(n)
  s = 1
  for i = 1 to n
      s = s + 2

Algoritme loop3(n)
  i = 0
  while i ≤ n^2
      j = 0
      while j ≤ i
          j = j + 1
      i = i + n

Algoritme loop4(n)
  i = 1
  j = 1
  while i ≤ n
      while j < i
          j = 2 · j
      i = 3 · i
```

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

	$\Theta(n \log n)$	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n^3)$	$\Theta(n^4)$	$\Theta((\log n)^2)$	$\Theta(n\sqrt{n})$	$\Theta(n^2)$	
loop1	A	B	C	D	E	F	G	H	2.1
loop2	A	B	C	D	E	F	G	H	2.2
loop3	A	B	C	D	E	F	G	H	2.3
loop4	A	B	C	D	E	F	G	H	2.4

Opgave 3 (Indsættelse i søgetræer, 4 %)



Angiv i hvilke blade A–K i ovenstående ubalancerede binære søgetræ elementerne 30, 12, 5, 23 og 3 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(30)	A	B	C	D	E	F	G	H	I	J	K	3.1
INSERT(12)	A	B	C	D	E	F	G	H	I	J	K	3.2
INSERT(5)	A	B	C	D	E	F	G	H	I	J	K	3.3
INSERT(23)	A	B	C	D	E	F	G	H	I	J	K	3.4
INSERT(3)	A	B	C	D	E	F	G	H	I	J	K	3.5

Opgave 4 (Heap-Extract-Max, 4 %)

1	2	3	4	5	6	7	8	9	10	11	12	13
26	25	22	17	12	21	6	16	14	1	8	4	18

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

1	2	3	4	5	6	7	8	9	10	11	12	
25	17	22	16	12	21	6	14	1	8	4	18	A
25	22	18	14	21	17	16	12	1	8	4	6	B
25	17	22	16	12	21	6		14	1	8	4	18	C
25	17	22	16	12	21	6	18	14	1	8	4	D
25	18	22	17	12	21	6	16	14	1	8	4	E

Opgave 5 (Build-Max-Heap, 4 %)

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

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

1	2	3	4	5	6	7	8	9	
8	9	6	7	2	1	3	5	4	A
1	2	3	4	5	6	7	8	9	B
9	7	8	5	2	1	6	3	4	C
9	8	6	5	7	1	3	2	4	D
9	8	7	6	5	4	3	2	1	E

Opgave 6 (Partition, 4 %)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
22	8	17	6	30	23	1	28	12	7	25	5	4	20	3

Angiv resultatet af at anvende PARTITION$(A, 4, 12)$ på ovenstående array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
22	8	17	1	5	23	6	28	12	7	25	30	4	20	3	A
1	3	4	5	6	7	8	12	17	20	22	23	25	28	30	B
22	8	17	1	5	6	7	12	23	25	28	30	4	20	3	C
22	8	17	1	5	6	30	23	28	12	7	25	4	20	3	D

Opgave 7 (Randomized-Quicksort, 4 %)

Angiv antal gange RANDOMIZED-QUICKSORT kalder PARTITION på et input af størrelse n.

	$\Theta(1)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	
Forvented antal kald til PARTITION	A	B	C	D	E	7.1
Færreste antal kald til PARTITION (best case)	A	B	C	D	E	7.2
Fleste antal kald til PARTITION (worst case)	A	B	C	D	E	7.3

Opgave 8 (Lineær probing, 4 %)

0	1	2	3	4	5	6	7	8	9	10
22	15					13	2			18

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

Angiv positionerne de fem elementer 0, 4, 6, 7 og 8 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 2, 13, 15, 18 og 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)	A	B	C	D	E	F	G	H	I	J	K	8.1
INSERT(4)	A	B	C	D	E	F	G	H	I	J	K	8.2
INSERT(6)	A	B	C	D	E	F	G	H	I	J	K	8.3
INSERT(7)	A	B	C	D	E	F	G	H	I	J	K	8.4
INSERT(8)	A	B	C	D	E	F	G	H	I	J	K	8.5

Opgave 9 (Dobbelt hashing, 4 %)

0	1	2	3	4	5	6	7	8	9	10
11		18		3			19			14

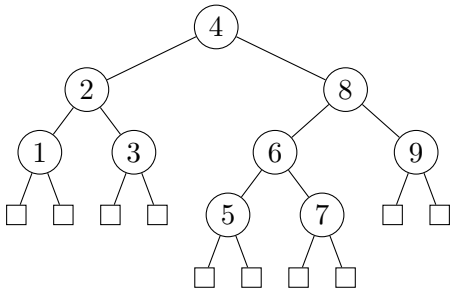
I ovenstående hashtabel af størrelse 11 er anvendt *dobbelt hashing* med hashfunktionerne $h_1(k) = 5k \bmod 11$ og $h_2(k) = 1 + (3k \bmod 10)$.

Angiv positionerne de fem elementer 0, 1, 2, 7 og 8 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 3, 11, 14, 18 og 19).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)	A	B	C	D	E	F	G	H	I	J	K	9.1
INSERT(1)	A	B	C	D	E	F	G	H	I	J	K	9.2
INSERT(2)	A	B	C	D	E	F	G	H	I	J	K	9.3
INSERT(7)	A	B	C	D	E	F	G	H	I	J	K	9.4
INSERT(8)	A	B	C	D	E	F	G	H	I	J	K	9.5

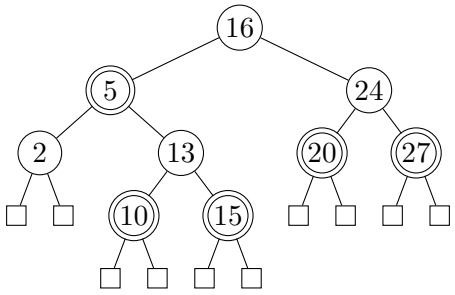
Opgave 10 (Rød-sorte træer, 4 %)

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

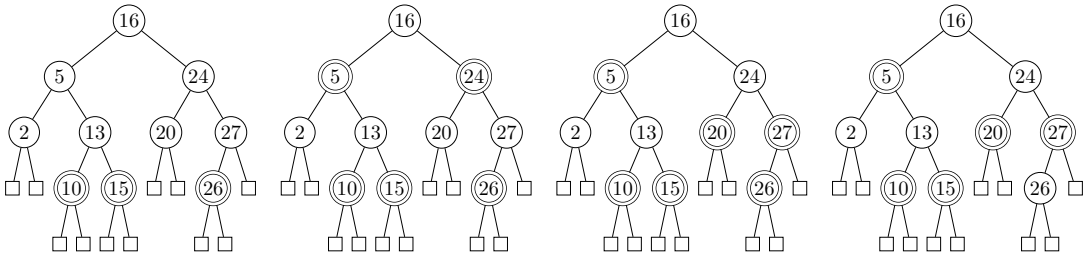


	Ja	Nej	
5, 7	A	B	10.1
4, 5, 7	A	B	10.2
2, 5, 7, 8	A	B	10.3
1, 3, 5, 7, 8	A	B	10.4
2, 5, 6, 7, 9	A	B	10.5

Opgave 11 (Indsættelse i rød-sorte træer, 4 %)



Angiv det resulterende rød-sorte træ når man indsætter 26 i ovenstående rød-sorte træ (dobbeltcirkler angiver røde knuder).

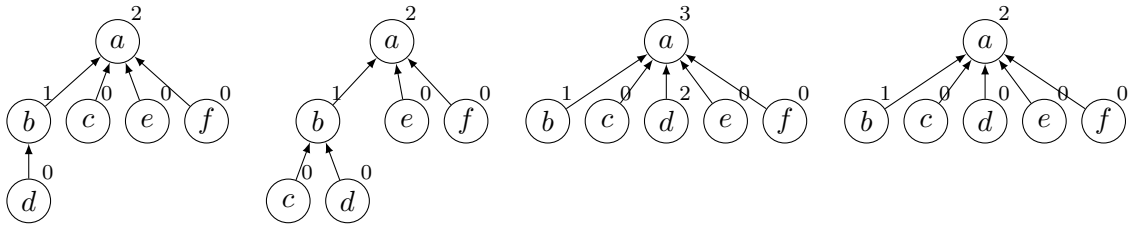


A B C D

Opgave 12 (Union-find, 4 %)

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



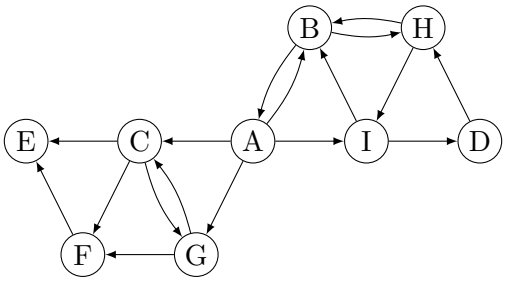
A B C D

Opgave 13 (Rekursionsligninger, 4 %)

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

	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2 \log n)$	$\Theta(n^3)$	
$T(n) = T(n-1) + n$	A	B	C	D	E	F	G	13.1
$T(n) = 2 \cdot T(n/3) + n$	A	B	C	D	E	F	G	13.2
$T(n) = 2 \cdot T(n/2) + n$	A	B	C	D	E	F	G	13.3
$T(n) = 4 \cdot T(n/2) + n^2$	A	B	C	D	E	F	G	13.4
$T(n) = 3 \cdot T(n/5) + n$	A	B	C	D	E	F	G	13.5

Opgave 14 (BFS, 4 %)

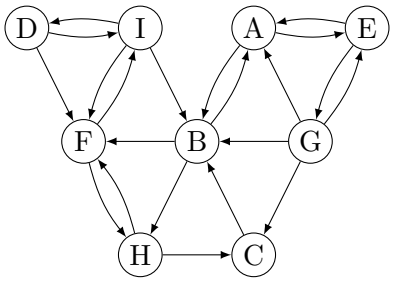


For et bredde først gennemløb (BFS) af ovenstående graf **startende i knuden A**, angiv rækkefølgen knuderne bliver udtaget af køen Q i BFS-algoritmen. Det antages, at grafen er givet ved incidenslister, hvor incidenslisterne er sorteret alfabetisk.

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

A B C D

Opgave 15 (DFS, 4 %)



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 **discovery time**.

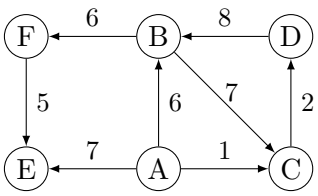
A B F H C I D E G A B H F I D C E G A B E F H G I C D A E G B F I D H C

A B C D 15.1

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

	Tree edge	Back edge	Cross edge	Forward edge	
(B, A)	A	B	C	D	15.2
(B, H)	A	B	C	D	15.3
(G, B)	A	B	C	D	15.4
(I, D)	A	B	C	D	15.5

Opgave 16 (Dijkstras algoritme, 4 %)

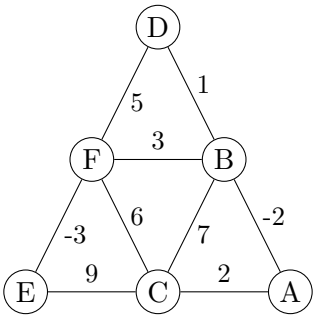


Antag Dijkstras algoritme anvendes til at finde korteste afstande fra **knuden A** til alle knuder i ovenstående graf. Angiv hvilken rækkefølge knuderne bliver taget ud af prioritetskøen i Dijkstras algoritme.

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

A B C D

Opgave 17 (Kruskals algoritme, 4 %)



Antag Kruskals algoritme anvendes til at finde et minimum udspændende træ for ovenstående graf. Angiv hvilken kant **sidst** inkluderes i det mininmum udspændende træ.

(E , F) (A , C) (B , D) (C , E) (B , F)

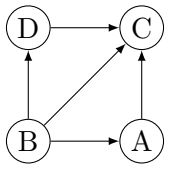
A B C D E

Opgave 18 (DFS på stærk sammenhængende grafer, 4 %)

Antag DFS udføres på en stærk sammenhængende graf med $n \geq 3$ knuder og m kanter, d.v.s. alle knuder udgør én stærk sammenhængskomponent. Det antages desuden at hvis (u, v) er en kant i grafen, så er (v, u) ikke en kant i grafen. Angiv hvor mange kanter af hver DFS type der maksimalt kan være.

	0	1	$n-1$	n	$m-n$	$m-n+1$	
Største antal træ kanter (tree edges)	A	B	C	D	E	F	18.1
Største antal baglæns kanter (back edges)	A	B	C	D	E	F	18.2
Største antal forlæns kanter (forward edges)	A	B	C	D	E	F	18.3
Største antal kryds kanter (cross edges)	A	B	C	D	E	F	18.4

Opgave 19 (Topologisk sortering, 4 %)



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

	Ja	Nej	
C D A B	A	B	19.1
B A D C	A	B	19.2
A D B C	A	B	19.3
B D A C	A	B	19.4
D A B C	A	B	19.5

Opgave 20 (Invarianter, 4 %)

Givet et ikke-negative heltal n, så beregner nedenstående algoritme $\lfloor\sqrt{n}\rfloor$.

Algoritme HeltalsKvadratrod(n)
Inputbetingelse : Heltal $n \geq 0$
Outputkrav : $low = \lfloor\sqrt{n}\rfloor$
Metode : $low \leftarrow 0$
$high \leftarrow n + 1$
$\{I\}$ **while** $low + 1 < high$ **do**
$m \leftarrow \lfloor(low + high)/2\rfloor$
if $m * m \leq n$ **then**
$low \leftarrow m$
else
$high \leftarrow m$

For hvert af følgende udsagn, angiv om det er en invariant I for algoritmen Heltals-Kvadratrod.

	Ja	Nej	
$0 \leq low < n$	A	B	20.1
$low \leq high$	A	B	20.2
$low < high$	A	B	20.3
$low^2 < n$	A	B	20.4
$n < high^2$	A	B	20.5

Dynamisk programmering

Antag vi har m heltallige møntværdier $c_1 < c_2 < \cdots < c_m$, hvor $c_1 = 1$. Givet et heltal n, ønsker vi at finde et heltal k, hvor $0 \leq k \leq n$, som vil kræve flest mønter at opnå. En møntværdi må bruges et vilkårligt antal gange. For mønterne $c_1 = 1, c_2 = 2, c_3 = 5, c_4 = 10, c_5 = 20$ og $n = 50$, vil $k = 39$ kræve 5 mønter ($39 = 2 + 2 + 5 + 10 + 20$). Alle andre værdier fra 0 til 50 kan også opnås med højst 5 mønter.

Vi lader T_i betegne det færreste antal mønter, der skal til for at opnå værdien i, hvor i er et ikke negativt heltal, som kan bestemmes ved følgende rekursionsformel:

$$T_i = \begin{cases} 0 & \text{hvis } i = 0 \\ \min\{1 + T_{i-c_j} \mid 1 \leq j \leq m \wedge c_j \leq i\} & \text{hvis } i > 0 \end{cases}$$

De følgende 3 opgaver består i at udfylde 3 blokke i følgende algoritmeskabelon.

Algoritme Mønter$(n, (c_1, \ldots, c_m))$
Opret tabel $T[0\,..\,n]$

for ...
≪ **Opgave 21**: iterer over T ≫

≪ **Opgave 22**: beregn $T[i]$ ≫

≪ **Opgave 23**: Udskriv et k der kræver flest mulige mønter≫

Opgave 21 (4 %)

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

	Ja	Nej	
for $i = 0$ **to** n	A	B	21.1
for $i = n$ **to** 0 **step** -1	A	B	21.2
for $i = 1$ **to** m	A	B	21.3

Opgave 22 (4 %)

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

	Ja	Nej	

$T[i] = 0$
if $i > 0$ **then**
  for $j = 1$ **to** m
  if $c_j \leq i$ **then**
  $T[i] = \min(T[i], 1 + T[i - c_j])$
	A	B	22.1

if $i = 0$ **then**
  $T[i] = 0$
else
  $T[i] = +\infty$
  for $j = 1$ **to** m
  if $c_j \leq i$ **then**
  $T[i] = \min(T[i], 1 + T[i - c_j])$
	A	B	22.2

if $i = 0$ **then**
  $T[i] = 0$
else
  $T[i] = +\infty$
  $j = 1$
  while $j \leq m \wedge c_j \leq i$ **do**
  $T[i] = \min(T[i], 1 + T[i - c_j])$
  $j = j + 1$
	A	B	22.3

Opgave 23 (4 %)

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

	Ja	Nej	
print $T[n]$	A	B	23.1

for $k = 1$ **to** n
  if $T[k] = m$ **do**
  print k
	A	B	23.2

$k = 0$
for $i = 1$ **to** n
  if $T[i] > T[k]$ **do**
  $k = i$
print k
	A	B	23.3

Opgave 24 (Amortiseret analyse, 4 %)

Betragt et rød-sort søgetræ, der gemmer en mængde reelle tal, udvidet med en operation DominatingInsert(x, δ), som indsætter tallet x i søgetræet og sletter alle eksisterende tal fra søgetræet i intervallet $[x-\delta, x+\delta]$. Dette implementeres ved at man gentagne gange søger efter forgængeren og efterfølgeren til x i søgetræet, og sletter de fundne tal hvis de ligger i intervallet $[x-\delta, x+\delta]$. Hvis indsættelsen sletter k tal, så tager dette worst-case $O((k+1)\log n)$ tid, hvor n er antal tal i træet før DominatingInsert udføres.

Angiv for hver af nedenstående funktioner om det er en potentialefunktion, hvormed man kan argumentere for at DominatingInsert tager amortiseret $O(\log n)$ tid.

	Ja	Nej	
$(k+n)\log n$	A	B	24.1
$n \log n$	A	B	24.2
$2n \log n$	A	B	24.3
$n \log k$	A	B	24.4
$k \log n$	A	B	24.5