EKSAMEN

Algoritmer og Datastrukturer

Mandag 9. januar 2023, 13:00–15:00

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 notation, 6 %)

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

	Ja	Nej	
$(\log n)^5/6$ er $O((\log n)^3)$	[A] 12%	[B] 80%	1.1
$n \cdot \log n$ er $O(n^{0.1})$	[A] 6%	[B] 86%	1.2
$\sqrt{n}$ er $O(n^{0.1})$	[A] 9%	[B] 83%	1.3
$n^{0.1} + \sqrt{n}$ er $O(\sqrt{n} \cdot \log n)$	[A] 75%	[B] 11%	1.4
$\log n$ er $O(n)$	[A] 93%	[B] 3%	1.5
$3(\log n)^6$ er $O(4^{\log n})$	[A] 81%	[B] 10%	1.6
$n^{1/3}$ er $O(n)$	[A] 91%	[B] 4%	1.7
$\sqrt{n} \cdot \log n$ er $O(\sqrt{n})$	[A] 0%	[B] 93%	1.8
$5n\sqrt{n} + \sqrt{n}$ er $O(\sqrt{n} \cdot \log n)$	[A] 5%	[B] 88%	1.9
$\log(n!) + n^2/3$ er $\Omega(n)$	[A] 87%	[B] 6%	1.10
$\log(n!) + n$ er $\Theta(n \cdot \log n)$	[A] 49%	[B] 34%	1.11
$\sqrt{n}$ er $\Omega(2^{3 \log n})$	[A] 5%	[B] 86%	1.12

Opgave 2 (Analyse af løkker, 6 %)

```
Algoritme loop1(n)
  s = 1
  while s ≤ n
    s = s + 1
```

```
Algoritme loop2(n)
  s = 0
  for i = 1 to n
    for j = 1 to i * i
      s = s + 1
```

```
Algoritme loop3(n)
  for i = 1 to n
    j = 1
    while j ≤ n
      j = 2 * j
```

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

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

	$\Theta(n^2)$	$\Theta(n)$	$\Theta((\log n)^2)$	$\Theta(n^3)$	$\Theta(\sqrt[3]{n})$	$\Theta(n \log n)$	$\Theta(\sqrt{n})$	$\Theta(\log n)$	
loop1	[A] 0%	[B] 96%	[C] 0%	[D] 0%	[E] 0%	[F] 0%	[G] 1%	[H] 0%	2.1
loop2	[A] 9%	[B] 0%	[C] 0%	[D] 81%	[E] 0%	[F] 1%	[G] 0%	[H] 0%	2.2
loop3	[A] 1%	[B] 2%	[C] 0%	[D] 0%	[E] 0%	[F] 84%	[G] 0%	[H] 4%	2.3
loop4	[A] 67%	[B] 8%	[C] 0%	[D] 2%	[E] 0%	[F] 1%	[G] 0%	[H] 0%	2.4

Opgave 3 (Max-Heap-Insert, 4 %)

Angiv den binære max-heap efter indsættelse af elementerne 6, 12, 9, 11, 14, 2 og 8 i den givne rækkefølge med MAX-HEAP-INSERT, startende med den tomme heap.

1	2	3	4	5	6	7	
12	14	9	11	6	2	8	A 0%
6	12	9	11	14	2	8	B 0%
14	12	9	11	6	2	8	C 3%
14	12	11	9	8	6	2	D 1%
14	12	9	6	11	2	8	☒ 95%

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

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

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

1	2	3	4	5	6	7	8	9	
6	8	9	7	4	5	2	1	3	A 0%
9	8	7	6	5	4	3	2	1	B 0%
9	7	8	6	4	5	2	1	3	C 6%
1	2	3	4	5	6	7	8	9	D 0%
9	8	6	7	4	5	2	1	3	☒ 91%

Opgave 5 (Rekursionsligninger, 4 %)

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

	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2 \log n)$	$\Theta(n^3)$	
$T(n) = 9 \cdot T(n/3) + 2$	A 1%	B 2%	C 0%	D 0%	☒ 73%	F 1%	G 10%	5.1
$T(n) = T(n-1) + \log n$	A 2%	B 0%	C 2%	☒ 86%	E 0%	F 1%	G 0%	5.2
$T(n) = 4 \cdot T(n/5) + n^2$	A 0%	B 0%	C 0%	D 0%	☒ 84%	F 4%	G 0%	5.3
$T(n) = T(n-2) + n^2$	A 0%	B 0%	C 0%	D 0%	E 3%	F 3%	☒ 78%	5.4
$T(n) = 3 \cdot T(n/9) + 2$	A 1%	☒ 82%	C 2%	D 1%	E 2%	F 0%	G 0%	5.5

Opgave 6 (Lineær probing, 4 %)

0	1	2	3	4	5	6	7	8	9	10
0	22	11							16	5

I ovenstående hashtabel af størrelse 11 er anvendt *lineær probing* med hashfunktionen $h(k) = 4k \bmod 11$.

Angiv positionerne de fem elementer 3, 4, 6, 8 og 9 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 0, 5, 11, 16 og 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(3)	A	B	C	~~D~~	E	F	G	H	I	J	K	6.1
	0%	0%	0%	98%	0%	0%	0%	0%	0%	0%	0%	
INSERT(4)	A	B	C	D	E	~~F~~	G	H	I	J	K	6.2
	0%	0%	0%	0%	0%	97%	0%	0%	0%	0%	0%	
INSERT(6)	A	B	C	~~D~~	E	F	G	H	I	J	K	6.3
	0%	0%	1%	95%	1%	0%	0%	0%	0%	0%	0%	
INSERT(8)	A	B	C	~~D~~	E	F	G	H	I	J	K	6.4
	0%	0%	1%	94%	0%	0%	0%	0%	0%	0%	0%	
INSERT(9)	A	B	C	~~D~~	E	F	G	H	I	J	K	6.5
	0%	0%	1%	96%	0%	0%	0%	0%	0%	0%	0%	

Opgave 7 (Dobbelt hashing, 4 %)

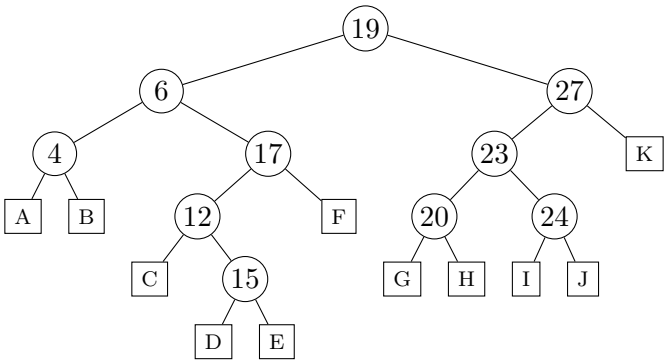
0	1	2	3	4	5	6	7	8	9	10
0					9	13			14	2

I ovenstående hashtabel af størrelse 11 er anvendt *dobbelt hashing* med hashfunktionerne $h_1(k) = 3k \bmod 11$ og $h_2(k) = 1 + (2k \bmod 10)$.

Angiv positionerne de fem elementer 3, 4, 5, 10 og 11 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 0, 2, 9, 13 og 14).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(3)	A	~~B~~	C	D	E	F	G	H	I	J	K	7.1
	0%	84%	3%	1%	1%	1%	0%	5%	0%	0%	0%	
INSERT(4)	A	~~B~~	C	D	E	F	G	H	I	J	K	7.2
	0%	95%	0%	0%	0%	0%	0%	0%	1%	0%	0%	
INSERT(5)	A	B	C	D	~~E~~	F	G	H	I	J	K	7.3
	0%	0%	0%	0%	94%	0%	0%	2%	0%	0%	0%	
INSERT(10)	A	B	C	D	E	F	G	H	~~I~~	J	K	7.4
	0%	3%	0%	0%	0%	0%	0%	1%	92%	0%	0%	
INSERT(11)	A	B	C	~~D~~	E	F	G	H	I	J	K	7.5
	0%	6%	4%	84%	0%	0%	0%	0%	0%	0%	0%	

Opgave 8 (Indsættelser i søgetræer, 4 %)

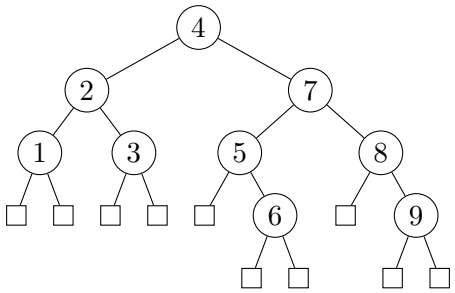


Angiv i hvilke blade A–K i ovenstående ubalancerede binære søgetræ elementerne 7, 14, 16, 11 og 22 skal indsættes (det antages at før hver indsættelse indeholder træet kun ovenstående ti elementer).

	A	B	C	D	E	F	G	H	I	J	K	
Insert(7)	0%	1%	**98%**	0%	0%	0%	0%	0%	0%	0%	0%	8.1
Insert(14)	0%	0%	0%	**99%**	0%	0%	0%	0%	0%	0%	0%	8.2
Insert(16)	0%	0%	0%	0%	**100%**	0%	0%	0%	0%	0%	0%	8.3
Insert(11)	0%	0%	**100%**	0%	0%	0%	0%	0%	0%	0%	0%	8.4
Insert(22)	0%	0%	0%	0%	0%	0%	0%	**99%**	0%	0%	0%	8.5

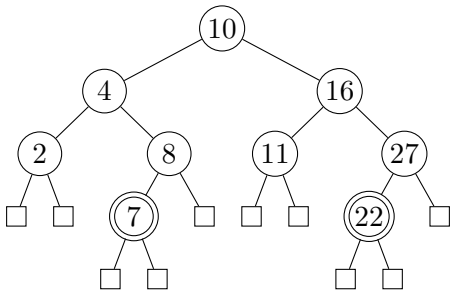
Opgave 9 (Rød-sorte træer, 4 %)

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

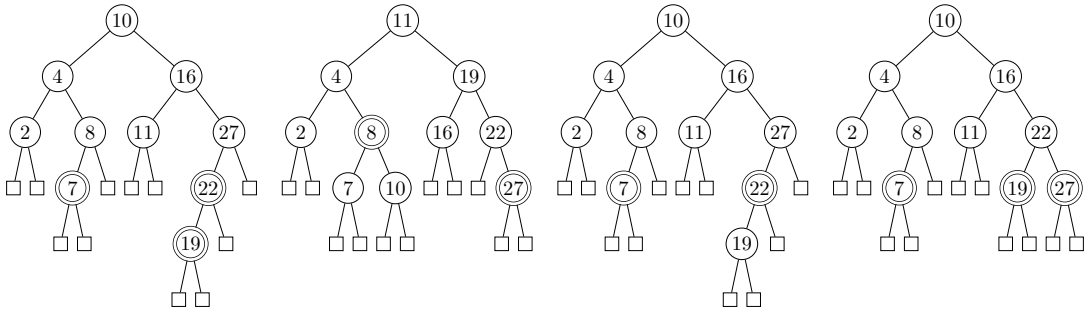


	Ja	Nej	
2, 6, 7, 9	**98%**	1%	9.1
1, 3, 5, 6, 8, 9	0%	**100%**	9.2
6, 9	**98%**	1%	9.3
1, 3, 6, 7, 9	**95%**	2%	9.4
4, 6, 9	1%	**98%**	9.5

Opgave 10 (Indsættelse i rød-sorte træer, 4 %)



Angiv det resulterende rød-sorte træ når man indsætter 19 i ovenstående rød-sorte træ (dobbeltcirkler angiver røde knuder).



A
0%

B
0%

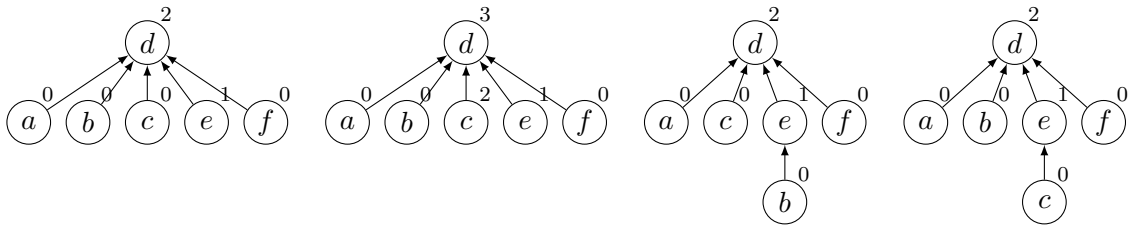
C
0%

☒
97%

Opgave 11 (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(b, e)
UNION(b, c)
UNION(a, d)
UNION(c, d)
UNION(f, c)
FIND-SET(b)



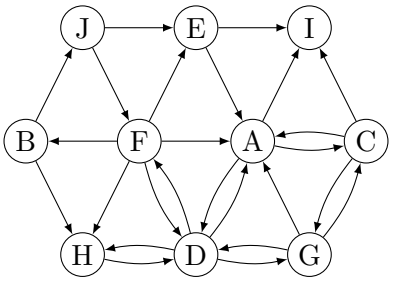
☒
66%

B
4%

C
16%

D
4%

Opgave 12 (BFS, 4 %)

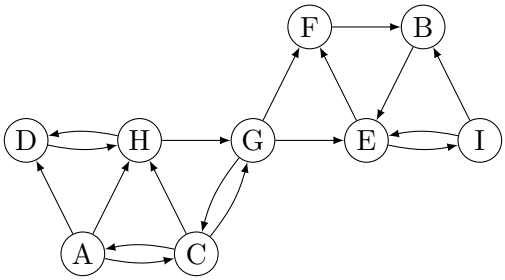


For et bredde først gennemløb (BFS) af ovenstående graf **startende i knuden A**, angiv rækkefølgen knuderne bliver indsat i køen Q i BFS-algoritmen. Det antages, at grafen er givet ved incidenslister, hvor incidenslisterne er sorteret alfabetisk.

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

A	B	C	☒
0%	1%	1%	96%

Opgave 13 (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 **finishing time**.

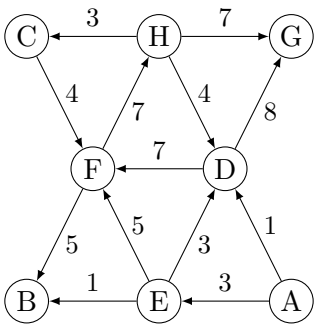
B I F E G H D C A D B F I E G H C A B F I E G D H C A D H I B F E G C A

A	B	☒	D	13.1
3%	1%	86%	5%	

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

	Tree edge	Back edge	Cross edge	Forward edge	
(C, A)	A 6%	☒ 87%	C 0%	D 0%	13.2
(I, B)	A 4%	B 1%	☒ 85%	D 0%	13.3
(C, H)	☒ 84%	B 0%	C 2%	D 6%	13.4
(A, H)	A 0%	B 0%	C 3%	☒ 87%	13.5

Opgave 14 (Dijkstras algoritme, 4 %)

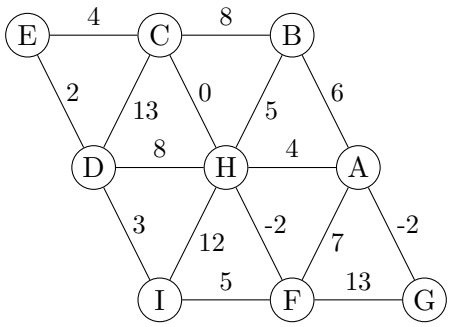


Antag Dijkstras algoritme anvendes til at finde korteste afstande fra **knuden A** til alle knuder i ovenstående graf. Angiv hvilken rækkefølge knuderne bliver taget ud af prioritetskøen i Dijkstra's algoritme.

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

☒ 89% B 2% C 0% D 6%

Opgave 15 (Prims algoritme, 4 %)

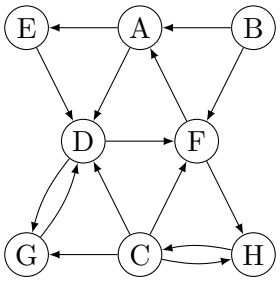


Antag Prims algoritme anvendes til at finde et minimum udspændende træ for ovenstående graf, og algoritmen starter i **knuden A**. Angiv hvilken rækkefølge knuderne bliver inkluderet i det minimum udspændende træ (taget ud af prioritetskøen i Prims algoritme).

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

☒ 89% B 1% C 1% D 6%

Opgave 16 (Stærke sammenhængskomponenter, 4 %)

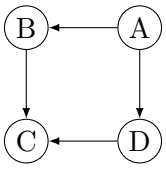


Hvad er antallet af stærke sammenhængskomponenter i ovenstående graf?

1 2 3 4 5 6 7 8

A	B	C	D	E	F	G	H
1%	78%	4%	3%	3%	0%	0%	0%

Opgave 17 (Topologisk sortering, 4 %)



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

	Ja	Nej	
A D B C	96%	1%	17.1
C B D A	0%	97%	17.2
A D C B	2%	94%	17.3
A B D C	96%	0%	17.4
C D B A	0%	97%	17.5

Opgave 18 (Select, 4 %)

Angiv worst-case og forventede tid for Randomized-Select og Deterministic-Select.

	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	
Forventede tid, Randomized-Select	A 5%	7%	C 8%	D 0%	18.1
Worst-case tid, Randomized-Select	A 0%	B 6%	C 1%	17%	18.2
Forventede tid, Deterministic-Select	A 5%	5%	C 6%	D 0%	18.3
Worst-case tid, Deterministic-Select	A 1%	8%	C 4%	D 6%	18.4

Opgave 19 (Amortiseret analyse, 4 %)

Antag vi anvender et rød-sort søgetræ til at gemme en mængde positive heltal, udvidet med en operation InsertOrDouble(x), som indsætter x i det rød-sorte søgetræ, hvis x ikke allerede er indsat. Ellers, slettes x fra søgetræet og InsertOrDouble($2 \cdot x$) kaldes rekursivt. F.eks., hvis vi udfører InsertOrDouble(3) på et rød-sort søgetræ, der gemmer mængden $\{1, 2, 3, 6, 12, 16, 25\}$, vil den resulterende mængde være $\{1, 2, 16, 24, 25\}$.

Angiv for hver af nedenstående funktioner om det er en potentialefunktion, hvormed man kan argumentere for at InsertOrDouble tager amortiseret $O(\log n)$ tid, hvor n er antal heltal i mængden før operationen.

	Ja	Nej	
$\log n$	[A] 4%	[X] 37%	19.1
n	[A] 6%	[X] 29%	19.2
$n \log n$	[X] 30%	[B] 4%	19.3
n^2	[A] 2%	[X] 33%	19.4
$\sum_{i=1}^{n} \log i$	[X] 29%	[B] 3%	19.5
$\sum_{i=1}^{n} i$	[A] 6%	[X] 27%	19.6

Opgave 20 (Invarianter, 4 %)

Givet to ikke-negative heltal x og y, så beregner nedenstående algoritme $x + y$.

Algoritme Addition(x, y)
Inputbetingelse : Heltal $x \geq 0$ og $y \geq 0$
Outputkrav : $r = x_0 + y_0$
Metode : $r \leftarrow 0$
$\{I\}$ **while** $x > 0$ **or** $y > 0$ **do**
if $x > 0$ **then**
$r \leftarrow r + 1$
$x \leftarrow x - 1$
else
$r \leftarrow r + 1$
$y \leftarrow y - 1$

For hvert af følgende udsagn, angiv om det er en invariant I for algoritmen Addition, hvor x_0 og y_0 angiver værdierne for henholdsvis x og y i starten.

	Ja	Nej	
$r = x + y$	[A] 4%	[X] 82%	20.1
$r \leq x + y$	[A] 16%	[X] 70%	20.2
$r = x_0 + y_0$	[A] 5%	[X] 79%	20.3
$r + x + y = x_0 + y_0$	[X] 78%	[B] 4%	20.4
$r + x_0 + y_0 = x + y$	[A] 3%	[X] 79%	20.5

Dynamisk programmering

De næste tre opgaver vedrører at løse *grenopsavnings* problemet ved hjælp af dynamisk programmering. Antag at vi har en gren af længde n centimeter, og vi skal save denne i mindre stykker, alle af længde $\leq m$ centimeter, ved at bruge mindst mulig energi. Lad e_i betegne energien, der skal bruges til at save grenen over efter i centimeter. Lad E_i være den minimale energi der skal til for at save de første i centimeter af grenen i stykker af længde $\leq m$, som kan bestemmes ved følgende rekursionsformel:

$$E_i = \begin{cases} 0 & \text{hvis } 0 \leq i \leq m \\ \min_{i-m \leq j < i}(E_j + e_j) & \text{hvis } m < i \leq n \end{cases}$$

Antag vi har en gren af længde $n = 7$, med nedenstående energiforbrug e_i for at save grenen over efter i centimeter. Bemærk $e_0 = e_n = 0$, da der ikke skal saves noget af.

i	0	1	2	3	4	5	6	7
e_i	0	7	1	4	4	2	5	0

For $m = 4$ vil det kræve mindst energi, hvis grenen saves over efter 2 og 5 centimer, som vil kræve energi $1 + 2 = 3$, og de tre resulterende stykker har længde 2, 3 og 2 centimer.

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

Algoritme Grenopsavning$(m, (e_0, \ldots, e_n))$
Opret tabel $E[0\,..\,n]$

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

≪ **Opgave 22**: beregn $E[i]$ ≫

≪ **Opgave 23**: Udskriv en løsning der kræver minimal energi ≫

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	☒ 62%	☐ B 4%	21.1
for $i = n$ **to** 0 **step** -1	☐ A 19%	☒ 45%	21.2

Opgave 22 (4 %)

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

	Ja	Nej	

$E[i] = 0$
if $i > m$ **then**
  for $j = i - m$ **to** $i - 1$
  $E[i] = \min(E[i], e_j + E[j])$
	A
	12%
		☒
		45%
			22.1

if $i \leq m$ **then**
  $E[i] = 0$
else
  $E[i] = \infty$
  for $j = i - m$ **to** $i - 1$
  $E[i] = \min(E[i], e_j + E[j])$
	☒
	51%
		B
		5%
			22.2

if $i \leq m$ **then**
  $E[i] = 0$
else
  $E[i] = \infty$
  for $j = i - 1$ **to** $i - m$ **step** -1
  $E[i] = \min(E[i], e_j + E[j])$
	☒
	37%
		B
		16%
			22.3

Opgave 23 (4 %)

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

	Ja	Nej	

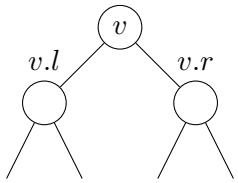
$i = n$
while $i > m$ **do**
  $j = i - 1$
  while $E[i] \neq e_j + E[j]$ **do**
  $j = j - 1$
  $i = j$
  print i
	☒
	30%
		B
		8%
			23.1

$i = E[n]$
while $i > m$ **do**
  print i
  $i = E[i]$
	A
	1%
		☒
		45%
			23.2

$i = n$
$j = n - 1$
while $j > 0$ **do**
  if $E[i] = e_j + E[j]$ **do**
  $i = j$
  print i
  $j = j - 1$
	☒
	34%
		B
		6%
			23.3

Opgave 24 (Udvidede søgetræer, 4 %)

Betragt et rød-sort søgetræ hvor hver knude gemmer et par af heltal (*element*, *vægt*), og parrene er sorteret fra venstre-mod-højre efter stigende *element* værdi. For en knude v i træet lader vi $v.e$ og $v.w$ betegne parret (e, w) gemt i knuden. Desuden gemmer v værdien $v.total$ som er summen af vægtene i alle knuder i v's undertræ.



Angiv hvorledes $v.total$ kan beregnes når den tilsvarende information er kendt ved de to børn $v.l$ og $v.r$ (det kan antages at disse begge eksisterer).

$v.total = \max(v.l.total, v.w, v.r.total)$ A 0%

$v.total = v.l.total + v.w + v.r.total$ ☒ 64%

$v.total = \max(v.l.total, v.l.total + v.w, v.l.total + v.w + v.r.total)$ C 3%

$v.total = \max(v.l.total + v.w, v.r.total + v.w)$ D 3%