EKSAMEN

Algoritmer og Datastrukturer

Mandag 6. januar 2025, 9:00–11:00

Institut for Datalogi, Naturvidenskabelige Fakultet, Aarhus Universitet

Antal sider i opgavesættet (incl. forsiden): 16

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	
$n \log n + n\sqrt{n}$ er $O(n^2)$	[X]	[B]	1.1
$7n \cdot \log n$ er $O(n^{2/3} \cdot n^{1/3})$	[A]	[X]	1.2
$n^{2/3}$ er $O(n \cdot \log n)$	[X]	[B]	1.3
$7 \log(n!)$ er $O(n^3)$	[X]	[B]	1.4
3^n er $O(n^3)$	[A]	[X]	1.5
$n^{0.1}$ er $O((\log n)^{10})$	[A]	[X]	1.6
$\sqrt{n} + 6n$ er $O(n \cdot \log n)$	[X]	[B]	1.7
n^2 er $O(\sum_{i=1}^{n} i)$	[X]	[B]	1.8
$(\log n)^2$ er $O(n)$	[X]	[B]	1.9
n^2 er $\Omega(2^{2 \log n})$	[X]	[B]	1.10
n^3 er $\Theta(3^{\log n})$	[A]	[X]	1.11
$\log n$ er $\Omega(n\sqrt{n})$	[A]	[X]	1.12

Opgave 2 (Analyse af løkker, 6 %)

```
Algoritme loop1(n)
  for i = 1 to n
      j = i
      while j ≥ 1
          j = ⌊j/2⌋
```

```
Algoritme loop2(n)
  s = 1
  for i = n to 1 step −1
      for j = i to 2 ∗ i
          s = s + 1
```

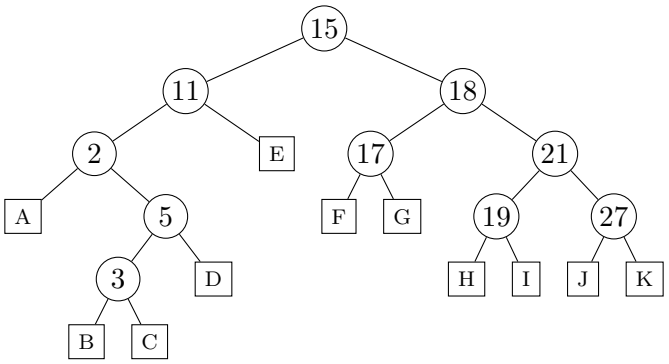
```
Algoritme loop3(n)
  i = 1
  j = n
  while i < j
      i = 2 ∗ i
      j = j + 1
```

```
Algoritme loop4(n)
  i = 0
  while i ∗ i ≤ n
      j = 0
      while j ≤ i
          j = j + 1
      i = i + 1
```

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

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

Opgave 3 (Indsættelse i søgetræer, 4 %)



Angiv i hvilke blade A–K i ovenstående ubalancerede binære søgetræ elementerne 6, 4, 26, 20 og 8 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(6)	A	B	C	~~D~~	E	F	G	H	I	J	K	3.1
INSERT(4)	A	B	~~C~~	D	E	F	G	H	I	J	K	3.2
INSERT(26)	A	B	C	D	E	F	G	H	I	~~J~~	K	3.3
INSERT(20)	A	B	C	D	E	F	G	H	~~I~~	J	K	3.4
INSERT(8)	A	B	C	~~D~~	E	F	G	H	I	J	K	3.5

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

Angiv den binære max-heap efter indsættelse af elementerne 13, 4, 11, 8, 10, 3 og 14 i den givne rækkefølge med MAX-HEAP-INSERT, startende med den tomme heap.

1	2	3	4	5	6	7	
14	10	13	4	8	3	11	~~X~~

1	2	3	4	5	6	7	
14	13	11	10	8	4	3	B

1	2	3	4	5	6	7	
14	10	13	8	4	3	11	C

1	2	3	4	5	6	7	
13	10	14	8	4	3	11	D

1	2	3	4	5	6	7	
13	4	11	8	10	3	14	E

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

1	2	3	4	5	6	7	8	9	10	11	12	13
26	25	22	15	21	18	7	12	6	11	13	14	16

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

1	2	3	4	5	6	7	8	9	10	11	12	13	
25	22	16	21	18	15	12	6	11	13	14	7		A
25	21	22	15	13	18	7	12	6	11		14	16	B
25	21	22	15	13	18	7	12	6	11	14	16		C
25	21	22	15	13	18	7	12	6	11	16	14		D
25	21	22	15	16	18	7	12	6	11	13	14		☒

Opgave 6 (Interval rapportering i binære søgetræer, 4 %)

Antag vi har et binært søgetræ indeholdende n elementer og træet har højde h (træet er ikke nødvendigvis balanceret). Nedenstående algoritme rapporterer alle elementer i intervallet $[\ell, r]$, som er indeholdt i et binært søgetræ med rod v.

```
Algoritme report(v, ℓ, r)
  if v ≠ Nil then
    if ℓ ≤ v.key then
      report(v.left, ℓ, r)
    if ℓ ≤ v.key ≤ r then
      print v.key
    if v.key ≤ r then
      report(v.right, ℓ, r)
```

Hvad er algoritmens worst-case udførselstid, når k betegner antallet af elementer der rapporteres?

$\Theta(k)$ $\Theta(k + \log n)$ $\Theta(k + h)$ $\Theta(k \cdot \log n)$ $\Theta(k \cdot h)$

A B ☒ D E

Opgave 7 (Lineær probing, 4 %)

0	1	2	3	4	5	6	7	8	9	10
0	22	8						21		18

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

Angiv positionerne de fem elementer 1, 2, 3, 7 og 11 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 0, 8, 18, 21 og 22).

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

Opgave 8 (Dobbelt hashing, 4 %)

0	1	2	3	4	5	6	7	8	9	10
		8		16			21	10	14	

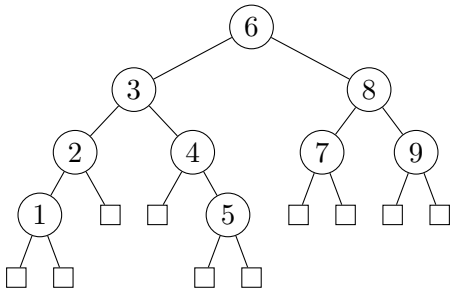
I ovenstående hashtabel af størrelse 11 er anvendt *dobbelt hashing* med hashfunktionerne $h_1(k) = 3k \bmod 11$ og $h_2(k) = 1 + (4k \bmod 10)$.

Angiv positionerne de fem elementer 2, 3, 5, 6 og 7 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 8, 10, 14, 16 og 21).

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

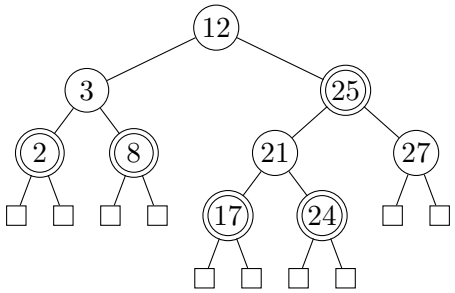
Opgave 9 (Rød-sorte træer, 4 %)

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

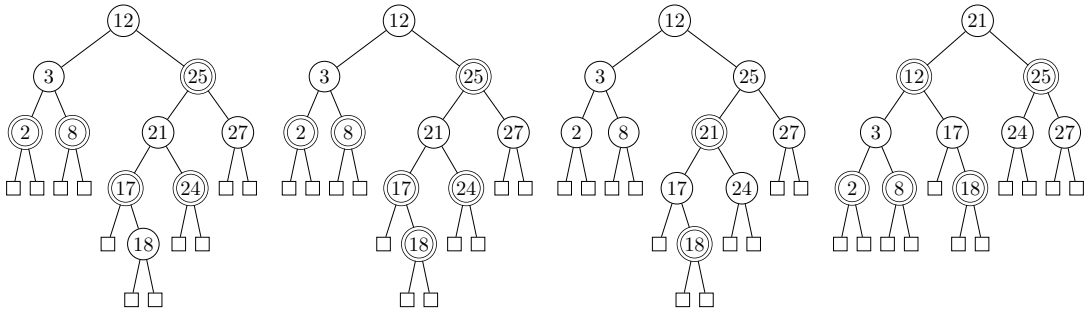


	Ja	Nej	
1, 5, 6	A	☒	9.1
1, 2, 4, 5, 7, 9	A	☒	9.2
1, 3, 5, 7, 9	☒	B	9.3
1, 5	☒	B	9.4
1, 3, 5, 8	☒	B	9.5

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



Angiv det resulterende rød-sorte træ når man indsætter 18 i ovenstående rød-sorte træ (dobbeltcirkler angiver røde knuder).



A B C ☒

Opgave 11 (Rekursionsligninger, 4 %)

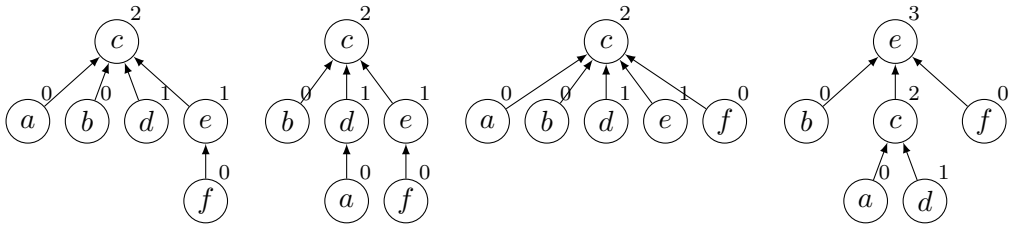
Angiv løsningen for hver af nedenstående rekursionsligninger, hvor $T(n) = 1$ for $n \leq 1$.

	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2 \log n)$	$\Theta(n^3)$	
$T(n) = 3 \cdot T(n/5) + n$	A	B	☒	D	E	F	G	11.1
$T(n) = 4 \cdot T(n/2) + 2$	A	B	C	D	☒	F	G	11.2
$T(n) = 4 \cdot T(n/5) + n$	A	B	☒	D	E	F	G	11.3
$T(n) = T(n-1) + n^2$	A	B	C	D	E	F	☒	11.4
$T(n) = T(\lfloor\sqrt{n}\rfloor) + n$	A	B	☒	D	E	F	G	11.5

Opgave 12 (Union-find, 4 %)

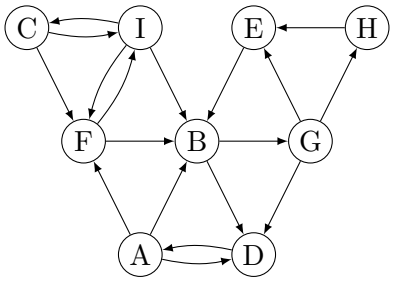
Angiv den resulterende union-find struktur efter nedenstående sekvens af operationer, når der anvendes union-by-rank og stikomprimering.

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



☒ B C D

Opgave 13 (BFS, 4 %)

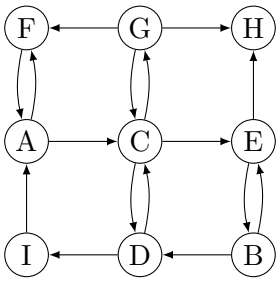


For et bredde først gennemløb (BFS) af ovenstående graf **startende i knuden A**, angiv rækkefølgen knuderne bliver indsat i køen Q i BFS-algoritmen. Det antages, at grafen er givet ved incidenslister, hvor incidenslisterne er sorteret alfabetisk.

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

[A] [B] [C] [☒]

Opgave 14 (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**.

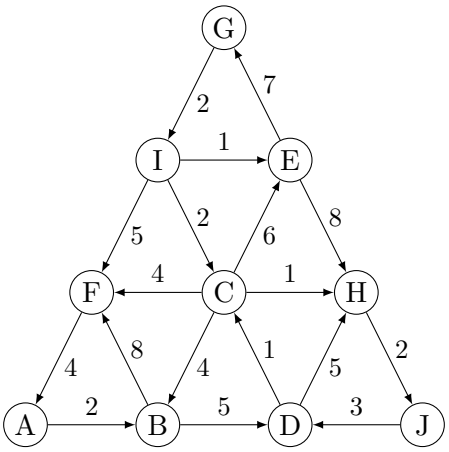
I D B H E F G C A H B I G E D F C A F I D B H E G C A F G H B E I D C A

[☒] [B] [C] [D] 14.1

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

	Tree edge	Back edge	Cross edge	Forward edge	
(B, D)	[A]	[B]	[☒]	[D]	14.2
(C, D)	[☒]	[B]	[C]	[D]	14.3
(A, F)	[A]	[B]	[C]	[☒]	14.4
(G, C)	[A]	[☒]	[C]	[D]	14.5

Opgave 15 (Dijkstras algoritme, 4 %)

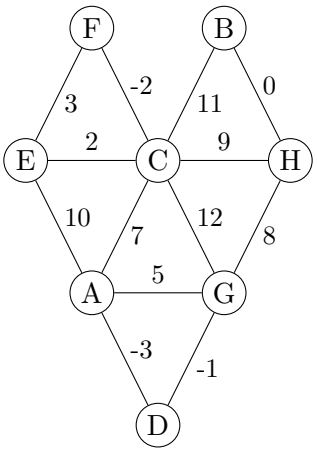


Antag Dijkstras algoritme anvendes til at finde korteste afstande fra **knuden A** til alle knuder i ovenstående graf. Angiv hvilken rækkefølge knuderne bliver taget ud af prioritetskøen i Dijkstras algoritme.

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

☐ A ☒ B ☐ C ☐ D

Opgave 16 (Kruskals algoritme, 4 %)

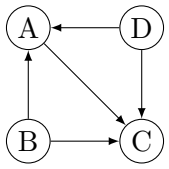


Antag Kruskals algoritme anvendes til at finde et minimum udspændende træ for ovenstående graf. Angiv hvilken kant **sidst** inkluderes i det minimum udspændende træ.

(A , C) (C , E) (B , H) (E , F) (G , H)

☐ A ☐ B ☐ C ☐ D ☒ E

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	
D A B C	A	☒	17.1
D B C A	A	☒	17.2
D B A C	☒	B	17.3
B D A C	☒	B	17.4
A D B C	A	☒	17.5

Opgave 18 (Korteste veje, 4 %)

Antag at Bellman-Ford algoritmen anvendes til at finde korteste veje fra en start knude til alle andre knuder i en graf. Hvad er worst-case antal RELAX operationer Bellman-Ford algoritmen kan lave på grafer med n knuder og $O(n)$ vægtede kanter?

$\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$ $\Theta(n^2 \log n)$ $\Theta(n^3)$

A B ☒ D E

Opgave 19 (Invarianter, 4 %)

Givet en *konkav* liste af n forskellige reelle tal $L[1..n]$, så beregner nedenstående algoritme maximum i listen. En liste er konkav hvis $L[i+1] - L[i] < L[i] - L[i-1]$, for $1 < i < n$.

F.eks. er listen $25, 100, 147, 174, 108, -56$ konkav.

Algoritme ConcaveMax($L[1..n]$)
Inputbetingelse : $n \geq 1$, alle $L[i] \in \mathbb{R}$ er forskellige, og
: $L[i+1] - L[i] < L[i] - L[i-1]$ for $1 < i < n$
Outputkrav : $L[low] = \max(L)$
Metode : $low \leftarrow 1$
$high \leftarrow n$
$\{I\}$ **while** $low < high$ **do**
$mid \leftarrow \lfloor (low + high)/2 \rfloor$
if $L[mid] < L[mid+1]$ **then**
$low \leftarrow mid + 1$
else
$high \leftarrow mid$

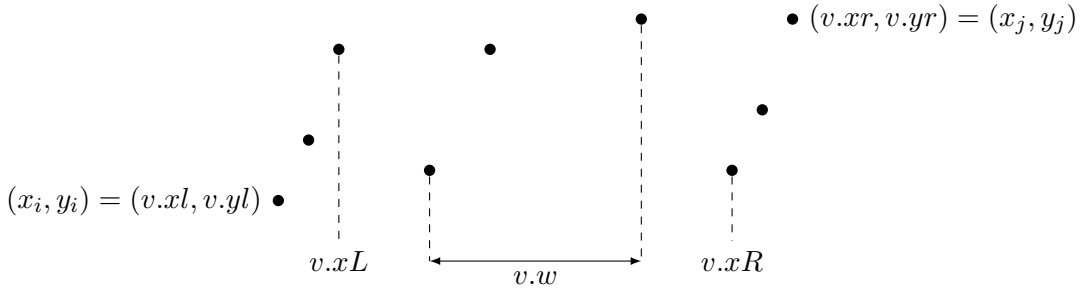
For hvert af følgende udsagn, angiv om det er en invariant I for algoritmen ConcaveMax, hvor $i_{\max}$ er indekset i L hvor $L[i_{\max}] = \max(L)$.

	Ja	Nej	
$low < high$	[A]	[X]	19.1
$low - 1 \leq high$	[X]	[B]	19.2
$low \leq i_{\max} \leq high$	[X]	[B]	19.3
$low < i_{\max} < high$	[A]	[X]	19.4
$1 \leq low \leq i_{\max} \leq high \leq n$	[X]	[B]	19.5

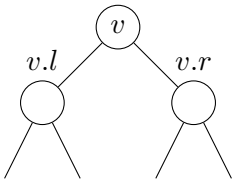
Opgave 20 (Udvidede søgetræer, 4 %)

Betragt n punkter $(x_1, y_1), \ldots, (x_n, y_n)$ i $\mathbb{R}^2$, hvor $x_1 < x_2 < \cdots < x_n$. En *sammenhængende delsekvens* af $j - i + 1$ punkter $(x_i, y_i), (x_{i+1}, y_{i+1}), \ldots, (x_j, y_j)$, hvor $1 \leq i \leq j \leq n$, siges at være *voksende* hvis og kun hvis $y_i < y_{i+1} < \cdots < y_j$ og *bredden* af delsekvensen er $x_j - x_i$. Vi ønsker at vedligeholde den maksimale bredde af en voksende sammenhængende delsekvens i en dynamisk mængde af punkter ved hjælp af et udvidet søgetræ.

Betragt et søgetræ hvor hver knude v gemmer et punkt $(v.x, v.y)$, og knuderne er ordnet venstre-mod-højre efter stigende $v.x$. Derudover gemmes i v reelle tal $v.w$, $v.xR$ og $v.xL$, $v.xl$, $v.yl$, $v.xr$ og $v.yr$. Hvis v's undertræ gemmer punkterne $(x_i, y_i), \ldots, (x_j, y_j)$, så er $(v.xl, v.yl) = (x_i, y_i)$, $(v.xr, v.yr) = (x_j, y_j)$, $v.w$ er den maksimale bredde af en sammenhængde voksende delsekvens af punkterne, $v.xR = x_s$ hvor (x_s, y_s) er det punkt længst til venstre, hvor $i \leq s \leq j$ og $(x_s, y_s), \ldots, (x_j, y_j)$ er voksende, og symmetrisk er $v.xL = x_t$ hvor (x_t, y_t) er det punkt længst til højre, hvor $i \leq t \leq j$ og $(x_i, y_i), \ldots, (x_t, y_t)$ er voksende. Bemærk at hvis alle punkterne udgør en voksende delsekvens, så er $v.xL = v.xr$ og $v.xR = v.xl$.



Angiv hvorledes $v.w$ kan beregnes når ovenstående information er kendt ved de to børn $v.l$ og $v.r$ (det kan antages at disse begge eksisterer). For to tal $a, b \in \mathbb{R}$ lader vi $\text{inc}(a, b) = 1$ hvis $a < b$, ellers er $\text{inc}(a, b) = 0$.



Hvilket af nedenstående udtryk beregner $v.w$?

$\max(v.l.w, v.r.w, v.r.xL - v.l.xR)$ A

$\max(v.l.w, v.r.w, \text{inc}(v.l.y, v.y) \cdot (v.r.xL - v.l.xL) + \text{inc}(v.y, v.r.y) \cdot (v.r.xR - v.l.xR))$ B

$\max(v.l.w, v.r.w, \text{inc}(v.l.yr, v.y) \cdot (v.x - v.l.xR) + \text{inc}(v.y, v.r.yl) \cdot (v.r.xL - v.x))$ ☒

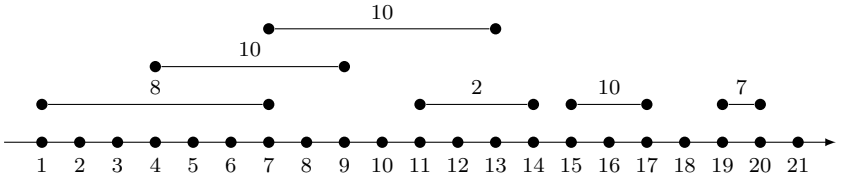
$\max(v.l.w, v.r.w, v.l.w + v.r.w + \text{inc}(v.l.y, v.y) \cdot (v.x - v.l.x) + \text{inc}(v.y, v.r.y) \cdot (v.r.x - v.x))$ D

Dynamisk programmering

Antag vi har n vægtede intervaller $(l_1, r_1, w_1), \ldots, (l_n, r_n, w_n)$, hvor det $i$te interval er $[l_i, r_i]$ med vægt w_i, og $l_i, r_i, w_i \in \mathbb{R}$ med $l_i < r_i$. Det antages at intervallerne er sorteret efter højre endepunkt, dvs. $r_1 \leq \cdots \leq r_n$. I denne opgave ønsker vi at finde en ikke overlappende delmængde af intervallerne, som har størst mulig samlet vægt vha. dynamisk programmering (overlap i endepunkterne er ikke tilladt).

F.eks. for de 6 intervaller

$$(1,7,8), (4,9,10), (7,13,10), (11,14,2), (15,17,10), (19,20,7)$$



opnår intervallerne $(4,9,10), (11,14,2), (15,17,10), (19,20,7)$ vægten 29.

Lad W_i betegne den maksimale vægt man kan opnå med en delmængde af de i første intervaller, hvor $0 \leq i \leq n$, som kan beregnes ved følgende rekursionsformel, hvor vi antager $r_0 = -\infty$:

$$W_i = \begin{cases} 0 & \text{hvis } i = 0 \\ \max(\{W_{i-1}\} \cup \{W_j + w_i \mid 0 \leq j < i \wedge r_j < l_i\}) & \text{hvis } 1 \leq i \leq n\,. \end{cases}$$

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

Algoritme DisjunkteIntervaller($(l_1, r_1, w_1), \ldots, (l_n, r_n, w_n)$))
Opret tabel $W[0\,..\,n]$ hvor alle indgange har værdien 0

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

≪ **Opgave 22**: beregn $W[i]$ ≫

≪ **Opgave 23**: Udskriv intervaller med maksimal vægt sum ≫

Opgave 21 (4 %)

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

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

Opgave 22 (4 %)

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

	Ja	Nej	

for $j = 0$ **to** $i - 1$
  if $W[j] + w_i > W[i]$ **then**
  $W[i] = \max(W[i-1], W[j] + w_i)$
	A	☒	22.1

$W[i] = W[i-1]$
for $j = 0$ **to** $i - 1$
  if $r_j < l_i$ **and** $W[j] + w_i > W[i]$ **then**
  $W[i] = W[j] + w_i$
	☒	B	22.2

$W[i] = W[i-1]$
for $j = i - 1$ **to** 0 **step** -1
  if $r_j < l_i$ **and** $W[j] + w_i > W[i]$ **then**
  $W[i] = W[j] + w_i$
	☒	B	22.3

Opgave 23 (4 %)

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

	Ja	Nej	

for $i = 1$ **to** n
  if $W[i-1] < W[i]$ **then**
  print (l_i, r_i)
	A	☒	23.1

for $i = n$ **to** 1 **step** -1
  if $W[i-1] < W[i]$ **then**
  print (l_i, r_i)
	A	☒	23.2

$z = W[n]$
for $i = n$ **to** 1 **step** -1
  if $W[i-1] < z$ **then**
  print (l_i, r_i)
  $z = z - w_i$
	☒	B	23.3

Opgave 24 (SUCCESSOR struktur, 4 %)

Antag vi ønsker at vedligeholde en datastruktur for en mængde af positive heltal S. I starten er $S = \{1, 2, \ldots, n\}$, som oprettes med CREATESET(n). Operationerne på S er DELETE(i), som fjerner i fra S, og SUCCESSOR(i), som returnerer det mindste tal i S der er større end eller lig med i. Vi antager at for Delete(i) er $i \in S$, og $1 \leq i < n$ for både DELETE(i) og SUCCESSOR(i), dvs. n slettes aldrig.

Vi antager operationerne er implementeret ved følgende kode, som vedligeholder et array $A[1..n]$. Hvis $i \in S$ er $A[i] = i$. Ellers er $i < A[i]$ og $j \notin S$ for $i \leq j < A[i]$.

```
proc CreateSet(n)
  for i = 1 to n do
    A[i] = i

proc Delete(i)
  A[i] = i + 1

proc Successor(i)
  if A[i] ≠ i then
    A[i] = Successor(A[i])
  return A[i]
```

Nedenstående er et eksempel på en datastruktur for mængden $\{1, 12, 15, 17, 19, 20\}$, efter en sekvens af operationer.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	**1**	12	7	5	11	7	12	11	10	11	12	**12**	15	15	**15**	17	**17**	19	**19**	**20**

Et kald til SUCCESSOR(4) vil returnere 12, og ændre $A[4] = 12$ og $A[5] = 12$.

Hvad er worst-case tiden for CREATESET(n) fulgt af en sekvens af en blanding af $n - 1$ DELETE og n SUCCESSOR operationer?

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

A B ☒ D E

I ovenstående betegner $\alpha(n)$ den meget langsomt voksende inverse Ackermann funktion, som indgår i analyses af union-find datastrukturer, der anvender stikomprimering og linking ved rang, hvor m union-find operationer på n elementer tager tid $O(m \cdot \alpha(n))$.