EKSAMEN

Algoritmer og Datastrukturer (520171E030)

Mandag 5. januar 2026, 10:00–12:00

Institut for Datalogi, Naturvidenskabelige Fakultet, Aarhus Universitet

Antal sider (incl. forsiden): **15**

Hjælpemidler/*Aids*: **Ingen/*None***

Kun det separate svarark skal udfyldes og afleveres.
Denne opgaveformulering skal *ikke* afleveres.

Only the separate answer sheet must be filled out and submitted.
This problem statement should* not *be submitted.

Vejledning og pointgivning

Dette eksamenssæt består af et antal 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 ét rigtigt svar. For hvert delspørgsmål må du vælge **max ét svar** ved at afkrydse den tilsvarende rubrik. Et delspørgsmål bedømmes som følger:

- Hvis du sætter kryds ved det rigtige svar, får du 1 point.
- Hvis du ikke sætter nogle 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 $s\,/\,n \cdot v\,\%$. Bemærk at det er muligt at få negative point for en opgave.

Information and Scoring

This exam set consists of a number of multiple-choice problems. The answers to the problems are indicated on the ***separate answer sheet*** *which is to be submitted. For each problem, the share of the total exam set is indicated. Each subproblem has exactly one correct answer. For each subproblem, you may choose* ***at most one answer*** *by crossing out the corresponding box. A subproblem is graded as follows:*

- *If you mark the correct answer, you get 1 point.*
- *If you do not mark any answers, you get 0 points.*
- *If you mark an incorrect answer, you get* $-\frac{1}{k-1}$ *points, where* k *is the number of answer options.*

For a problem with weight $v\,\%$ *and with* n *subproblems, where you achieve a total of* s *points, your score for the problem is calculated as* $s\,/\,n \cdot v\,\%$*. Note that it is possible to get negative points for a problem.*

Korrekte markeringer
Correct markings

1.1 [A] [X] [C] [D] [E]
1.2 [A] [B] [C] [■] [E]

Ugyldige markeringer
Invalid markings

2.1 [A] [B] [X] [X] [E]
2.2 [X] [B] [C] [D] [E]
2.3 [A] [B] [C] [D] [E]
2.4 [A] (B) [C] [D] [E]
2.5 [A] [B] [C] [D] [/]

En markering skal være et tydeligt mørkt kryds mellem hjørnerne af et felt, eller et fuldstændig udfyldt felt. Hvis man laver en fejl, kan man skrive ved siden af svarmulighederne hvad man svarer.

A marking must be a clear dark cross between the corners of a box, or a completely filled box. If you make a mistake, you can write next to the answer options what your answer.

Opgave/*Problem* 1 (Asymptotisk notation/*Asymptotic notation*, 6 %)

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

In the following, $\log n$ *denotes the base-2 logarithm of* n.

	Ja/*Yes*	Nej/*No*	
$n^2 = O(2^n)$	A	B	1.1
$3^n/2^n = O(n^2)$	A	B	1.2
$\log(n^2) = O(2^{\log n})$	A	B	1.3
$4n^{2/3} + 2^{3\log n} = O((\log n)^6)$	A	B	1.4
$2n \cdot \log n = O(3^n)$	A	B	1.5
$n \cdot \log n = O(n!)$	A	B	1.6
$(\log n)^3 = O(\sqrt{n})$	A	B	1.7
$n \cdot \log n + \sqrt{n}/5 = O(\sqrt{n})$	A	B	1.8
$3n \cdot \log n = O(n \cdot \log n)$	A	B	1.9
$\log n = \Omega(1)$	A	B	1.10
$6 + (\log n)^3 = \Omega(n^3)$	A	B	1.11
$2^{\log n} = \Theta(n)$	A	B	1.12

Opgave/*Problem* 2 (Analyse af løkker/*Analysis of loops*, 6 %)

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

```
Algorithm loop2(n)
  s = 1
  for i = 1 to n
    for j = 1 to i
      s = s + 1
```

```
Algorithm loop3(n)
  i = 0
  s = 0
  while s ≤ n * n
    i = i + 1
    s = s + i
```

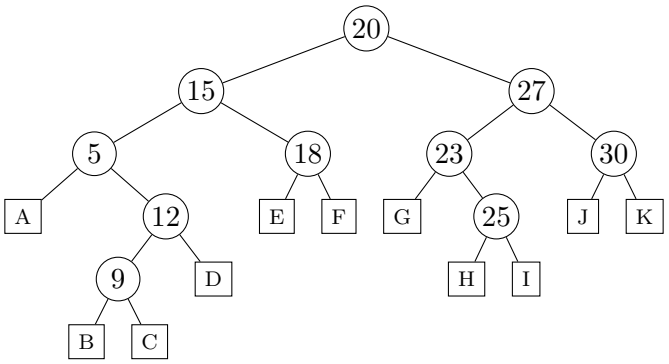
```
Algorithm loop4(n)
  i = 2
  while i ≤ n
    j = i
    while j ≤ n
      j = j + 1
    i = i * i
```

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

Indicate for each of the above algorithms the running time as a function of n *in* Θ*-notation.*

	$\Theta(n^2)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n^3)$	$\Theta(n \cdot \log\log n)$	$\Theta((\log n)^2)$	$\Theta(n \log n)$	$\Theta(\sqrt{n})$	
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/*Problem* 3 (Indsættelse i søgetræer/*Insertions into search trees*, 4 %)



Angiv i hvilke blade A–K i ovenstående ubalancerede binære søgetræ elementerne 4, 29, 31, 22 and 14 skal indsættes (det antages at før hver indsættelse indeholder træet kun ovenstående ti elementer).

Indicate in which leaves A–K in the above unbalanced binary search tree the elements 4, 29, 31, 22 and 14 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(4)	A	B	C	D	E	F	G	H	I	J	K	3.1
INSERT(29)	A	B	C	D	E	F	G	H	I	J	K	3.2
INSERT(31)	A	B	C	D	E	F	G	H	I	J	K	3.3
INSERT(22)	A	B	C	D	E	F	G	H	I	J	K	3.4
INSERT(14)	A	B	C	D	E	F	G	H	I	J	K	3.5

Opgave/*Problem* 4 (Max-Heap-Insert, 4 %)

Angiv den binære max-heap efter indsættelse af elementerne 8, 6, 2, 11, 7, 10 og 14 i den givne rækkefølge med MAX-HEAP-INSERT, startende med den tomme heap.

What is the binary max-heap after inserting the elements 8, 6, 2, 11, 7, 10 and 14 in the given order with MAX-HEAP-INSERT*, starting with the empty heap.*

1	2	3	4	5	6	7	
14	11	10	8	7	6	2	A
14	8	11	6	7	2	10	B
8	11	14	6	7	10	2	C
14	11	10	6	7	8	2	D
8	6	2	11	7	10	14	E

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

1	2	3	4	5	6	7	8	9	10	11	12	13
26	24	20	17	21	19	1	5	14	8	4	7	11

Angiv den resulterende max-heap efter at anvende HEAP-EXTRACT-MAX på ovenstående max-heap?

What is the resulting max-heap after applying HEAP-EXTRACT-MAX *to the above max-heap?*

1	2	3	4	5	6	7	8	9	10	11	12	
24	21	17	20	19	11	5	14	8	4	7	1	A

1	2	3	4	5	6	7	8	9	10	11	12	
24	21	20	17	8	19	1	5	14	11	4	7	B

1	2	3	4	5	6	7	8	9	10	11	12	
24	21	20	17	11	19	1	5	14	8	4	7	C

1	2	3	4	5	6	7	8	9	10	11	12	13	
24	21	20	17	8	19	1	5	14		4	7	11	D

1	2	3	4	5	6	7	8	9	10	11	12	
24	21	20	17	8	19	1	5	14	4	7	11	E

Opgave/*Problem* 6 (Merge-Sort, 4 %)

Algorithm MERGE-SORT(A, p, r)
  if $r - p + 1 \leq k$
    INSERTION-SORT(A, p, r)
  else
    $q = \lfloor (p + r)/2 \rfloor$
    MERGE-SORT(A, p, q)
    MERGE-SORT$(A, q + 1, r)$
    MERGE(A, p, q, r)

Betragt ovenstående variant af MERGE-SORT-algoritmen, som anvender INSERTION-SORT til at sortere delarrays af størrelse højst k. Angiv udførselstiden for denne algoritme i Θ-notation som funktion af både n og k, hvor n er størrelsen af det array der skal sorteres.

Consider the above variant of the MERGE-SORT *algorithm, which uses* INSERTION-SORT *to sort subarrays of size at most k. What is the running time of this algorithm in Θ-notation as a function of both n and k, where n is the size of the array to be sorted?*

$\Theta(n \log(nk))$ $\Theta(nk \log n)$ $\Theta(nk + n \log n)$ $\Theta(k^2 + n \log n)$ $\Theta(nk + n((\log n) - k))$

A B C D E

Opgave/*Problem* 7 (Lineær probing/*Linear probing*, 4 %)

0	1	2	3	4	5	6	7	8	9	10
			5			21	10		15	13

I ovenstående hashtabel af størrelse 11 er anvendt *lineær probing* med hashfunktionen $h(k) = 5k \bmod 11$. Angiv positionerne de fem elementer 2, 7, 8, 9 og 11 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 5, 10, 13, 15 og 21).

In the above hash table of size 11, linear probing *is used with hash function* $h(k) = 5k \bmod 11$. *Indicate the positions the five elements 2, 7, 8, 9 and 11 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 5, 10, 13, 15 and 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	7.1
Insert(7)	A	B	C	D	E	F	G	H	I	J	K	7.2
Insert(8)	A	B	C	D	E	F	G	H	I	J	K	7.3
Insert(9)	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/*Problem* 8 (Kvadratisk probing/*Quadratic probing*, 4 %)

0	1	2	3	4	5	6	7	8	9	10
11			22					6	4	13

I ovenstående hashtabel af størrelse 11 er anvendt *kvadratisk probing* med hashfunktionerne $h'(k) = 5k \bmod 11$ og $h(k,i) = (h'(k) + 2i + i^2) \bmod 11$. Angiv positionerne de fem elementer 1, 2, 5, 8 og 9 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 4, 6, 11, 13 og 22).

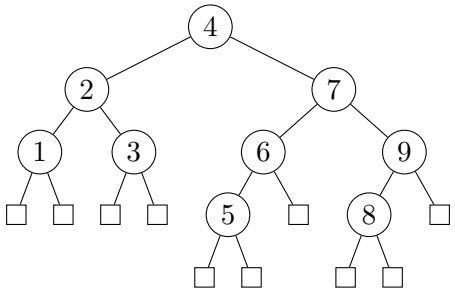
In the above hash table of size 11, quadratic probing *is used with hash functions* $h'(k) = 5k \bmod 11$ *and* $h(k,i) = (h'(k) + 2i + i^2) \bmod 11$. *Indicate the positions the five elements 1, 2, 5, 8 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 4, 6, 11, 13 and 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	8.1
Insert(2)	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(8)	A	B	C	D	E	F	G	H	I	J	K	8.4
Insert(9)	A	B	C	D	E	F	G	H	I	J	K	8.5

Opgave/*Problem* 9 (Lovlige rød-sorte træer/*Valid red-black trees*, 4 %)

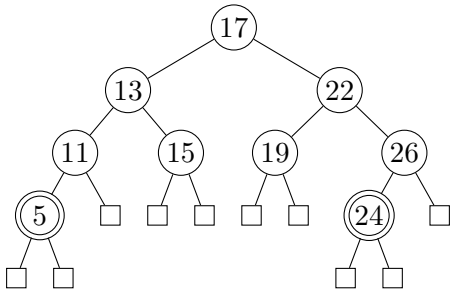
For hver af nedenstående delmængder, angiv om nedenstående binære træ er et lovligt rød-sort træ hvis disse knuder farves røde og resten af knuderne sorte.

For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red and all other nodes black.



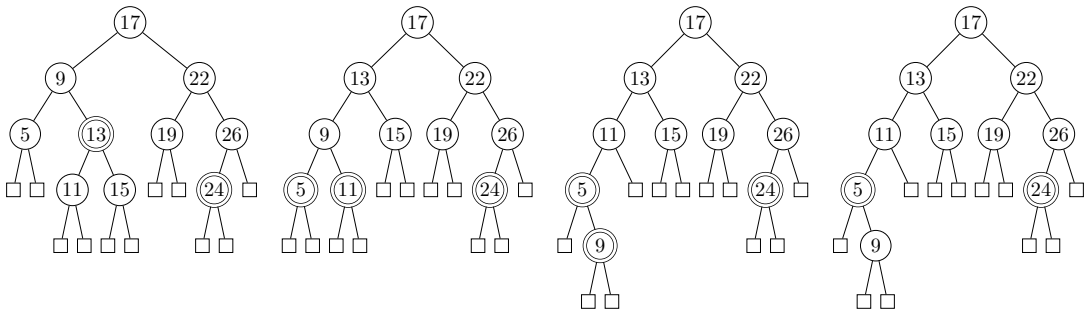
	Ja/*Yes*	Nej/*No*	
1, 3, 5, 6, 8, 9	A	B	9.1
4, 5, 8	A	B	9.2
1, 3, 5, 7, 8	A	B	9.3
2, 5, 7, 8	A	B	9.4
5, 8	A	B	9.5

Opgave/*Problem* 10 (Rød-sort træ indsættelse/*Red-black tree insertion*, 4 %)



Angiv det resulterende rød-sorte træ når man indsætter 9 i ovenstående rød-sorte træ (dobbeltcirkler angiver røde knuder).

Indicate the resulting red-black tree when inserting 9 into the above red-black tree (double circles indicate red nodes).



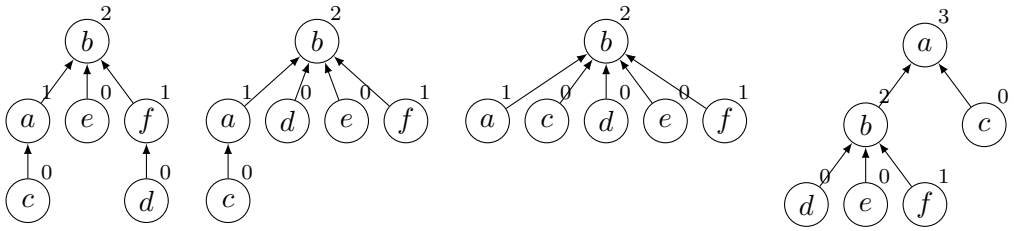
A B C D

Opgave/*Problem* 11 (Union-find, 4 %)

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

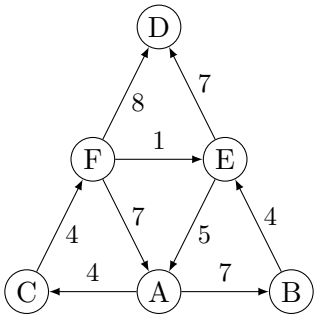
Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

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



A B C D

Opgave/*Problem* 12 (Dijkstras algoritme/*Dijkstra's algorithm*, 4 %)



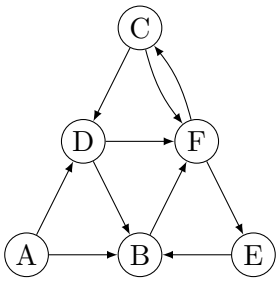
Antag Dijkstras algoritme anvendes til at finde korteste afstande fra **knuden A** til alle knuder i ovenstående graf. Angiv hvilken rækkefølge knuderne bliver taget ud af prioritetskøen i Dijkstras algoritme.

*Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.*

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

A B C D

Opgave/*Problem* 13 (BFS, 4 %)



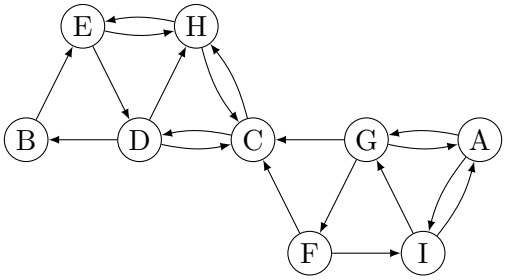
For en bredde først søgning (BFS) i ovenstående graf **startende i knuden A**, angiv rækkefølgen knuderne bliver udtaget af køen Q i BFS-algoritmen. Det antages, at grafen er givet ved alfabetisk sorterede incidenslister.

*For a breadth first search (BFS) on the graph above **starting in node A**, indicate the order in which the nodes are removed from the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.*

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

A B C D

Opgave/*Problem* 14 (DFS, 4 %)



Betragt en dybde først søgning (DFS) i ovenstående graf startende i **knuden A**, hvor de udgående kanter til en knude besøges i alfabetisk rækkefølge. Angiv i hvilken rækkefølge knuderne får tildelt **finishing time**.

*Consider a depth first search (DFS) on the graph above starting in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **finishing time**.*

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

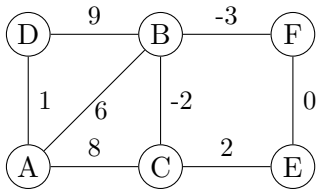
A B C D 14.1

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

Indicate for each of the edges below which type the edge is in the DFS search.

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

Opgave/*Problem* 15 (Kruskals algoritme/*Kruskal's algorithm*, 4 %)



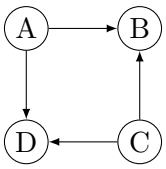
Antag Kruskals algoritme anvendes til at finde et minimum udspændende træ for ovenstående graf. Angiv hvilken kant **sidst** inkluderes i det mininmum udspændende træ.

*Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.*

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

A B C D E

Opgave/*Problem* 16 (Topologisk sortering/*Topological sorting*, 4 %)



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

For each of the node orderings below, state whether it is a valid topological sort of the above graph.

	Ja/*Yes*	Nej/*No*	
C A B D	A	B	16.1
A D B C	A	B	16.2
B A D C	A	B	16.3
A C D B	A	B	16.4
A C B D	A	B	16.5

Opgave/*Problem* 17 (Rekursionsligninger/*Recurrence relations*, 4 %)

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

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2 \log n)$	$\Theta(n^3)$	
$T(n) = 9 \cdot T(n/3) + 3$	A	B	C	D	E	F	G	17.1
$T(n) = T(n-1) + n^2$	A	B	C	D	E	F	G	17.2
$T(n) = 8 \cdot T(n/2) + n$	A	B	C	D	E	F	G	17.3
$T(n) = T(n/3) + 5$	A	B	C	D	E	F	G	17.4
$T(n) = T(n-2) + 1$	A	B	C	D	E	F	G	17.5

Opgave/*Problem* 18 (Sortering/*Sorting*, 4 %)

Givet et array $A[1..n]$, vil nedenstående algoritme sortere A når man kalder SortIt$(A, 1, n)$. Angiv algoritmens udførselstid som funktion af n.

Given an array $A[1..n]$, the below algorithm sorts A when calling SortIt$(A, 1, n)$. Indicate the algorithm's running time as a function of n.

```
Algorithm SortIt(A, i, j)
    if j = i + 1 and A[i] > A[j]
        swap A[i], A[j]
    else if j ≥ i + 2
        s = ⌊(j − i + 1)/3⌋
        SortIt(A, i, j − s)
        SortIt(A, i + s, j)
        SortIt(A, i, j − s)
```

$\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^{3/2})$ $\Theta(n^{\log_2 3})$ $\Theta(n^{\log_{3/2} 3})$ $\Theta(n^2)$

A B C D E F

Opgave/*Problem* 19 (Invarianter/*Invariants*, 4 %)

Nedenstående algoritme PairSum(A, x) afgør om en værdi x kan opnås som summen af to elementer $A[i] + A[j]$ fra et sorteret array $A[1..n]$ (muligvis det samme element to gange).

The below algorithm PairSum(A, x) determines if a value x can be obtained as the sum of two entries $A[i] + A[j]$ from a sorted array $A[1..n]$ (possibly the same entry twice).

```
Algorithm PairSum(A, x)
Input  : x, A[1..n]: A[1] ≤ ··· ≤ A[n]
Output : i, j: (1 ≤ i ≤ j ≤ n ∧ A[i] + A[j] = x) ∨
                (i > j ∧ (∀i', j' : 1 ≤ i' ≤ j' ≤ n ⇒ A[i'] + A[j'] ≠ x))
Method : i = 1
         j = n
         {I} while i ≤ j and A[i] + A[j] ≠ x
                   if x < A[i] + A[j] then
                       j = j − 1
                   else
                       i = i + 1
```

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

For each of the following statements, indicate whether it is an invariant I for the algorithm PairSum.

	Ja/*Yes*	Nej/*No*	
$1 \le i \le n$	A	B	19.1
$1 \le i \le j + 1 \le n + 1$	A	B	19.2
$A[i] + A[j] = x$	A	B	19.3
$\forall i', j' : 1 \le i' \le i \wedge j \le j' \le n \Rightarrow A[i'] + A[j'] \ne x$	A	B	19.4
$\forall i', j' : (1 \le i' < i \wedge 1 \le j' \le n) \vee (1 \le i' \le n \wedge j < j' \le n) \Rightarrow A[i'] + A[j'] \ne x$	A	B	19.5

Opgave/*Problem* 20 (Huffman koder/*Huffman codes*, 4 %)

Bogstav/*Letter*	a	b	c	d	e	f	g
Frekvens/*Frequency*	20	40	10	10	60	50	50

Antag vi har en fil med ovenstående bogstaver og frekvenser. Hvis man konstruerer et Huffman træ for disse frekvenser, hvor lange er koderne så for `a`, `c` og `e`?

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `a`*,* `c`*, and* `e`*?*

	1	2	3	4	5	6	
`a`	A	B	C	D	E	F	20.1
`c`	A	B	C	D	E	F	20.2
`e`	A	B	C	D	E	F	20.3

Filen indeholder 20 + 40 + 10 + 10 + 60 + 50 + 50 = 240 bogstaver. Hvor mange bits vil en Huffman kodning af strengen bruge?

The file contains 20 + 40 + 10 + 10 + 60 + 50 + 50 = 240 letters. How many bits would a Huffman encoding of the string use?

480 560 600 620 670 710

A B C D E F 20.4

Opgave/*Problem* 21 (Amortiseret analyse/*Amortized analysis*, 4 %)

Algorithm PUSH(Q, x)
  INSERT(I, x)

Algorithm REMOVE(Q, x)
  INSERT(D, x)
  if $2 * |D| > |I|$
    rebuild $I = I - D$, $D = \emptyset$

Algorithm POP(Q)
  while $|D| > 0 \wedge$ MIN(I)=MIN(D)
    EXTRACT-MIN(D)
    EXTRACT-MIN(I)
  $v =$ EXTRACT-MIN(I)
  if $2 * |D| > |I|$
    rebuild $I = I - D$, $D = \emptyset$
  return v

Vi ønsker at designe en prioritetskø, som understøtter de tre operationer PUSH(Q, x), POP(Q) og REMOVE(Q, x), som henholdvis indsætter et element x i Q, returnerer og fjerner det mindste element i Q, og fjerner et givet element x fra Q (som skal findes i Q).

Vi repræsenterer Q ved to binære heaps I og D, og implementerer operationerne som ovenfor, hvor *rebuild* operationen bygger en ny binær heap I indeholdende alle elementer i I som ikke er i D vha. sortering i tid $O(|I| \log |I|)$. Der mindes om at en binær heap H understøtter operationerne INSERT(H, x) og EXTRACT-MIN(H) i tid $O(\log |H|)$, og MIN(H) i tid $O(1)$.

Angiv for hver af nedenstående funktioner om det er en potentialefunktion, hvormed man kan argumentere for at hver af de tre operationer PUSH, POP og REMOVE tager amortiseret $O(\log n)$ tid, hvor n er antallet af elementer i Q.

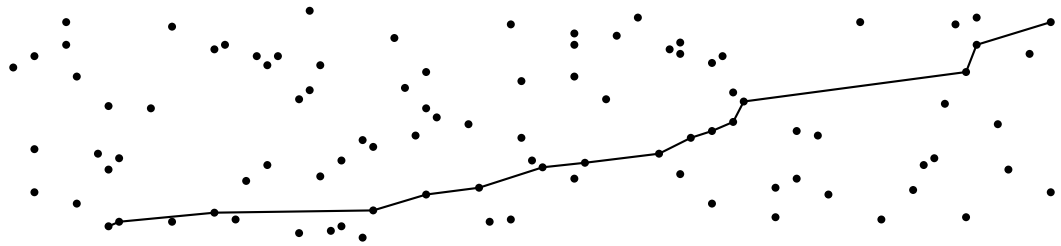
We wish to design a priority queue, which supports the three operations PUSH(Q, x), POP(Q) *and* REMOVE(Q, x)*, which respectively inserts an element x into Q, returns and removes the smallest element in Q, and removes a given element x from Q (which must be found in Q).*

We represent Q by two binary heaps I and D, and implement the operations as above, where the rebuild operation builds a new binary heap I containing all elements in I which are not in D using sorting in time $O(|I| \log |I|)$. Recall that a binary heap H supports the operations INSERT(H, x) *and* EXTRACT-MIN(H) *in time $O(\log |H|)$, and* MIN(H) *in time $O(1)$.*

For each of the following functions, indicate whether it is a potential function with which one can argue that each of the three operations PUSH, POP *and* REMOVE *take amortized $O(\log n)$ time, where n is the number of elements in Q.*

	Ja/*Yes*	Nej/*No*	
$\|I\| \log \|I\|$	A	B	21.1
$\|D\| \log \|D\|$	A	B	21.2
$\|D\| \log \|I\|$	A	B	21.3
$(\|I\| + \|D\|) \cdot \log(\|I\| + \|D\|)$	A	B	21.4
$(\|I\| - \|D\|) \cdot \log(\|I\| - \|D\|)$	A	B	21.5

Dynamisk programmering/*Dynamic programming*



Givet n punkter $(x_1, y_1), \ldots, (x_n, y_n)$, hvor $x_1 \leq \cdots \leq x_n$, ønsker vi at finde en delsekvens af punkterne, som stiger skarpt m.h.t. både x- og y-koordinaterne, og som indeholder flest mulige punkter. Hvis $L(i)$ betegner det maksimale antal punkter i en skarpt stigende delsekvens, som starter i (x_i, y_i), så kan $L(i)$ bestemmes ved hjælp af følgende rekursionsligning.

Given n points $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_1 \leq \cdots \leq x_n$, we want to find a subsequence of the points that is strictly increasing with respect to both the x- and y-coordinates, and contains the largest number of points. If $L(i)$ denotes maximum number of points in a strictly increasing subsequence starting at (x_i, y_i), then $L(i)$ can be determined using the following recurrence relation.

$$L(i) = \max\Big(\{1\} \cup \{1 + L(j) \mid i < j \leq n \ \wedge \ x_i < x_j \ \wedge \ y_i < y_j\}\Big)$$

De følgende 3 opgaver består i at udfylde 3 blokke i følgende algoritmeskabelon, hvor tabellen L gemmer ovenstående $L(i)$ værdier og J bruges til at rapportere en optimal løsning.

The following 3 problems consist of filling in 3 boxes in the following algorithm template, where L stores the above $L(i)$ values and J is used to construct an optimal solution.

Algorithm IncreasingPoints($(x_1, y_1), \ldots, (x_n, y_n)$)
Create table $L[1 \,..\, n]$, where all entries have the value 1
Create table $J[1 \,..\, n]$, where all entries have the value 0

for ...
≪ **Problem 22**: iterate over L ≫

≪ **Problem 23**: compute $L[i]$ ≫

≪ **Problem 24**: print longest increasing subsequence of points ≫

Opgave/*Problem* 22 (4 %)

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

For each of the code fragments below, indicate whether it could lead to a correct solution.

	Ja/*Yes*	Nej/*No*	
for $i = 1$ **to** n	A	B	22.1
for $i = n$ **to** 1 **step** -1	A	B	22.2

Opgave/*Problem* 23 (4 %)

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.
For each of the code fragments below, indicate whether it could lead to a correct solution.

	Ja/*Yes*	Nej/*No*	

for $j = i + 1$ **to** n
  if $x_i < x_j \ \wedge \ y_i < y_j \ \wedge \ L[i] < 1 + L[j]$ **then**
  $L[i] = 1 + L[j]$
  $J[i] = j$
	A	B	23.1

for $j = i + 1$ **to** n
  if $x_i < x_j \ \wedge \ y_i < y_j$ **then**
  $L[i] = \max(L[i], 1 + L[j])$
  $J[i] = j$
	A	B	23.2

for $j = n$ **to** $i + 1$ **step** -1
  if $x_i < x_j \ \wedge \ y_i < y_j \ \wedge \ L[i] \leq L[j]$ **then**
  $L[i] = 1 + L[j]$
  $J[i] = j$
	A	B	23.3

Opgave/*Problem* 24 (4 %)

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.
For each of the code fragments below, indicate whether it could lead to a correct solution.

	Ja/*Yes*	Nej/*No*	

$i = 1$
while $i > 0$
  print (x_i, y_i)
  $i = J[i]$
	A	B	24.1

$i = 1$
for $j = 2$ **to** n
  if $L[j] > L[i]$ **then**
  $i = j$
while $i > 0$
  print (x_i, y_i)
  $i = J[i]$
	A	B	24.2

$i = J[1]$
while $i > 0$
  print (x_i, y_i)
  $i = J[i]$
	A	B	24.3