



Hvilket af nedenstående svar gælder for følgende rekursionsligning?

$$T(n) = 5 \cdot T(n/2) + n^2$$

Selected Answer:  [None Given]

Answers:

 $T(n) = \Theta(n^p)$ med $p = \log_2(5)$.

$$T(n) = \Theta(n^p) \text{ med } p = \log_5(2).$$

$$T(n) = \Theta(n^p \log n) \text{ med } p = \log_2(5).$$

$$T(n) = \Theta(n^p \log n) \text{ med } p = \log_5(2) .$$

$$T(n) = \Theta(n^2).$$

Rekursionsligningen kan ikke løses med Master Theorem.

Hvilket af nedenstående svar gælder for følgende rekursionsligning?


$$T(n) = 5 \cdot T(n/5) + n \log n$$

Selected Answer:  [None Given]

Answers:


$$T(n) = \Theta(n^p) \text{ med } p = \log_5(5).$$

$$T(n) = \Theta(n^p \log n) \text{ med } p = \log_5(5).$$

-  Rekursionsligningen kan ikke løses med Master Theorem.

Hvilket af nedenstående svar gælder for følgende rekursionsligning?

$$T(n) = 2 \cdot T(n/5) + n^{1/2}$$

Selected Answer:  [None Given]


Answers:

$$T(n) = \Theta(n^p) \text{ med } p = \log_2(5).$$

$$T(n) = \Theta(n^p) \text{ med } p = \log_5(2).$$

$$T(n) = \Theta(n^p \log n) \text{ med } p = \log_2(5).$$

$$T(n) = \Theta(n^p \log n) \text{ med } p = \log_5(2).$$

 $T(n) = \Theta(n^{1/2}).$

Rekursionsligningen kan ikke løses med Master Theorem.

Hvilket af nedenstående svar gælder for følgende rekursionsligning?


$$T(n) = 2 \cdot T(n - 2) + n$$

Selected Answer:  [None Given]

Answers:

$$T(n) = \Theta(n^p) \text{ med } p = \log_2(2).$$

$$T(n) = \Theta(n^p \log n) \text{ med } p = \log_2(2).$$


 Rekursionsligningen kan ikke løses med Master Theorem.


Hvilke af nedenstående udsagn er sande? [*Et eller flere svar.*]

Selected Answers:  [None Given]

Answers:

2^n er $O(n^3)$


 n^2 er $O(3^n)$


 $n(\log n)^2$ er $O(n^3 \log n)$


$n^2 \log n$ er $O(n(\log n)^3)$

n^3 er $O(n^2)$

3^n er $O(2^n)$

 $n^{1/3}$ er $O(n^{1/2})$


 $(1/3)^n$ er $O((1/2)^n)$

 3 er $O(2)$

Udfør først $\text{HEAP-INCREASE-KEY}(A, 9, 15)$ og dernæst $\text{HEAP-EXTRACT-MAX}(A)$ på nedenstående max-heap A .

	1	2	3	4	5	6	7	8	9
A:	18	9	16	4	8	12	13	1	2

Hvilket af nedenstående svar angiver udseendet af heapen efter disse to operationer?

Selected Answer:  [None Given]

Answers:

	1	2	3	4	5	6	7	8
A:	16	15	13	9	8	4	12	1



	1	2	3	4	5	6	7	8
A:	16	15	13	9	8	12	4	1


	1	2	3	4	5	6	7	8
A:	16	15	13	4	8	12	2	1

	1	2	3	4	5	6	7	8
A:	16	15	13	4	8	12	1	2

Nedenstående er en hashtabel H der bruger quadratic probing, med auxiliary hashfunktion $h'(x) = (3x + 5) \bmod 11$ og med konstanter $c_1 = 3$ og $c_2 = 1$.

	0	1	2	3	4	5	6	7	8	9	10
H :	13	39		36					23	5	


Indsæt værdierne 22, 16 og 17 (i den rækkefølge). Hvilket af nedenstående svar angiver udseendet af hashtabellen efter disse tre operationer?

Selected Answer:  [None Given]

Answers:

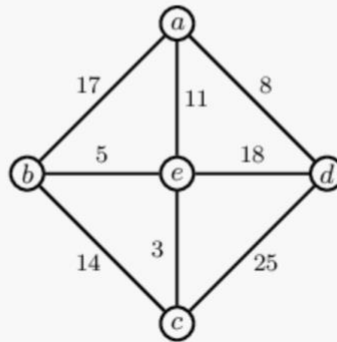
	0	1	2	3	4	5	6	7	8	9	10
H :	13	39	17	36		22			23	5	16

	0	1	2	3	4	5	6	7	8	9	10
H :	13	39	16	36	17	22			23	5	

	0	1	2	3	4	5	6	7	8	9	10
 H :	13	39	16	36		22		17	23	5	

	0	1	2	3	4	5	6	7	8	9	10
H :	13	39	16	36		22			23	5	17

Udfør Prims algoritme på grafen nedenfor, med start i knuden a . Den første knude, som udtages fra prioritetskøen (med operationen EXTRACT-MIN) under algoritmens kørsel, er knuden a . Hvilken knude er den sidste, som udtages?



Selected Answer: [None Given]

Answers:

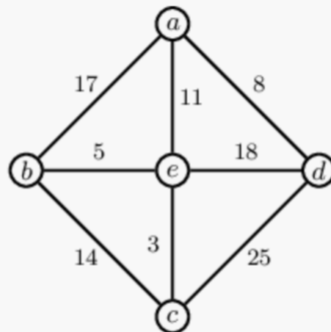
Knuden b

Knuden c

Knuden d

Knuden e

For den samme graf, hvilke af følgende kanter er med i det minimum spanning tree (MST) som Prims algoritme returnerer? [*Et eller flere svar.*]



Selected Answers: [None Given]

Answers:

Kanten (a, b)

Kanten (b, c)

Kanten (c, d)

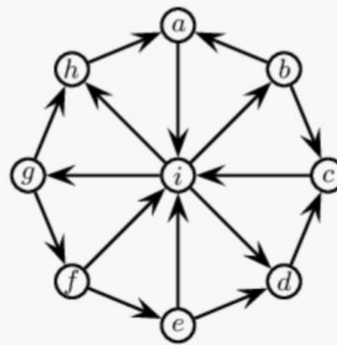
Kanten (d, a)

Kanten (b, e)

Kanten (e, d)

På grafen nedenfor, udfør dybde-først søgning (DFS) med start i knuden i . For DFS afhænger resultatet af ordningen af knuders nabolister. Du skal her antage at en knudes naboliste er sorteret i alfabetisk orden efter naboknuderne navne.

Hvilken knude får starttiden (discovery time) 12?



Selected Answer: [None Given]

Answers:

Knuden d

Knuden e

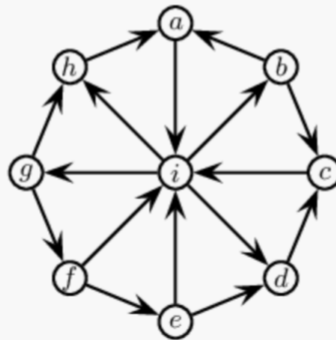
Knuden f

Knuden g

Ingen knude har starttid 12.

Vi ser stadig på dybde-først søgning (DFS) på denne graf med start i knuden i . Du skal stadig antage at en knudes naboliste er sorteret i alfabetisk orden efter naboknudernes navne.

Hvilken knude får sluttiden (finishing time) 16?



Selected Answer: [None Given]

Answers:

Knuden b

Knuden d

Knuden f

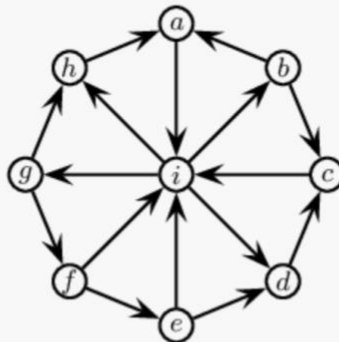
Knuden h

Ingen knude har sluttid 16.

Vi ser stadig på dybde-først søgning (DFS) på denne graf med start i knuden i . Du skal stadig antage at en knudes naboliste er sorteret i alfabetisk orden efter naboknudernes navne.

Bestem for alle kanter deres type (tree edge, back edge, forward edge, cross edge).

Hvilket af svarene nedenfor indeholder præcis én kant af hver type?



Selected Answer: ✖ [None Given]

Answers:

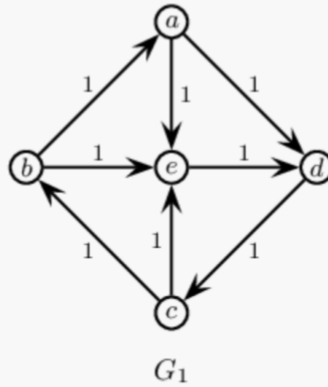
$(b, i), (a, b), (b, c), (d, i)$

✔ $(b, i), (a, i), (h, i), (c, d)$

$(g, h), (f, i), (d, e), (a, b)$

$(a, i), (h, i), (d, e), (e, i)$

Hvilke af nedenstående algoritmer kan bruges til at finde korteste veje på denne graf? [Et eller flere svar.]



Selected Answers: [None Given]

Answers:

DIJKSTRA

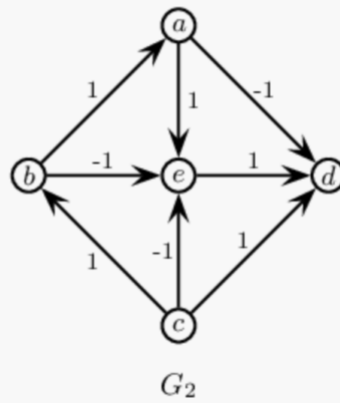
BELLMAN-FORD

DAG-SHORTEST-PATHS

BREATH-FIRST-SEARCH

DEPTH-FIRST-SEARCH

Hvilke af nedenstående algoritmer kan bruges til at finde korteste veje på denne graf? [*Et eller flere svar.*]



Selected Answers: ❌ [None Given]

Answers:

DIJKSTRA

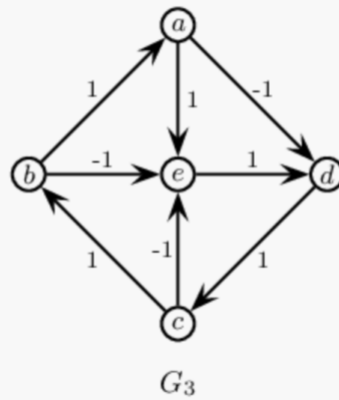
✔️ BELLMAN-FORD

✔️ DAG-SHORTEST-PATHS

BREATH-FIRST-SEARCH

DEPTH-FIRST-SEARCH

Hvilken af nedenstående algoritmer kan bruges til at finde korteste veje på denne graf?



Selected Answer: [None Given]

Answers:

DIJKSTRA

BELLMAN-FORD

DAG-SHORTEST-PATHS

BREATH-FIRST-SEARCH

DEPTH-FIRST-SEARCH


På grafer med n knuder og $m = n \log n$ kanter, hvad er den asymptotiske køretid som funktion af n for algoritmen DIJKSTRA (med prioritetskøen implementeret med en binær heap)?

Selected Answer:  [None Given]

Answers:

$O(n)$

$O(n \log n)$

 $O(n(\log n)^2)$

$O(n^2)$

$O(n^2 \log n)$

$O(n^2(\log n)^2)$

På grafer med n knuder og $m = n \log n$ kanter, hvad er den asymptotiske køretid som funktion af n for algoritmen BELLMAN-FORD?

Selected Answer:  [None Given]


Answers:

$O(n)$

$O(n \log n)$


$O(n(\log n)^2)$

$O(n^2)$

 $O(n^2 \log n)$


$O(n^2(\log n)^2)$

På grafer med n knuder og $m = n \log n$ kanter, hvad er den asymptotiske køretid som funktion af n for algoritmen DAG-SHORTEST-PATHS?

Selected Answer:  [None Given]

Answers:

$O(n)$

 $O(n \log n)$

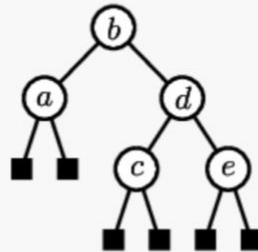
$O(n(\log n)^2)$

$O(n^2)$

$O(n^2 \log n)$

$O(n^2(\log n)^2)$

Hvilke farvning af knuderne i træet nedenfor gør det til et rød-sort træ? [Et eller flere svar.]



Selected Answers: [None Given]

Answers:

Sorte: b, d
Røde: a, c, e



Sorte: a, b, c, e
Røde: d

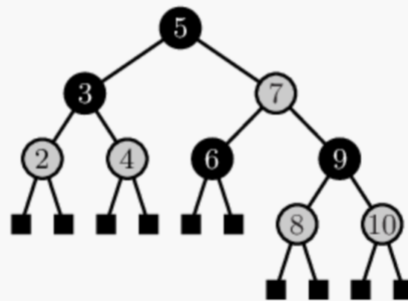


Sorte: a, b, d
Røde: c, e

Sorte: a, c, e
Røde: b, d

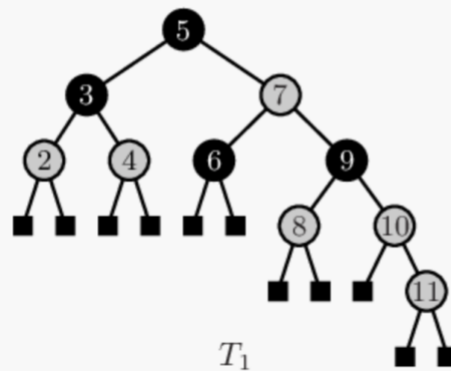
Sorte: a, b, c, d, e
Røde: ingen

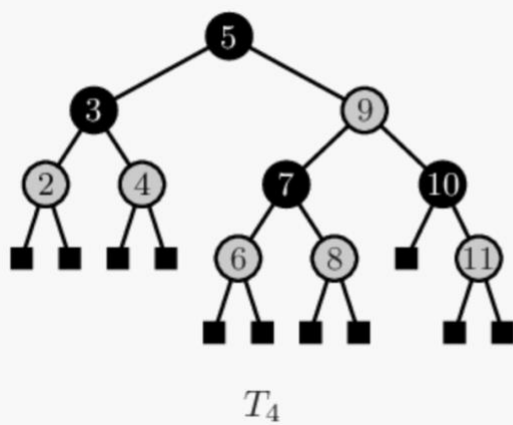
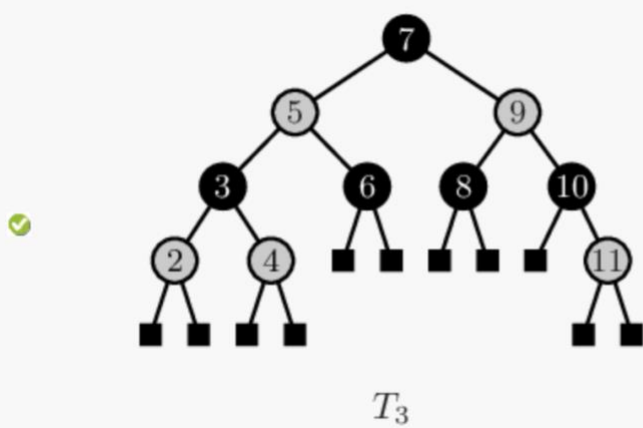
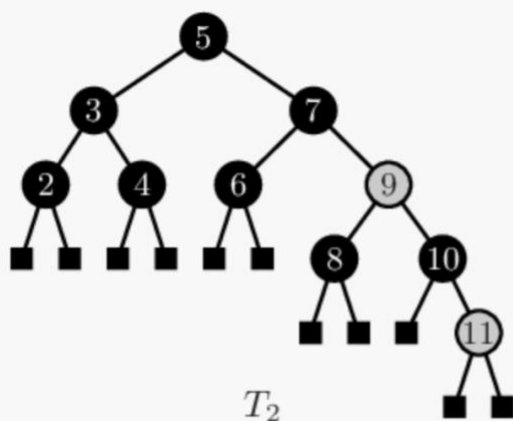
I nedenstående rød-sorter træ indsættes 11 under brug af algoritmen fra lærebogen. Hvordan ser træet ud efter indsættelsen?



Selected Answer: [None Given]


Answers:





For følgende algoritme, hvad er den asymptotiske køretid i Θ -notation som funktion af n ?

```
ALGORITME1( $n$ )  
   $i = n$   
  while  $i > 1$   
     $j = n$   
    while  $j > i$   
       $j = j - 1$   
     $i = i - 1$ 
```

Selected Answer:  [None Given]

Answers:

$\Theta(\log n)$

$\Theta(n)$

$\Theta(n \log n)$


 $\Theta(n^2)$

$\Theta(n^3)$

$\Theta(2^n)$


For følgende algoritme, hvad er den asymptotiske køretid i Θ -notation som funktion af n ?

```
ALGORITME2( $n$ )  
   $i = 1$   
   $j = 1$   
  while  $i \leq n$   
    while  $j \leq i$   
       $j = j + 1$   
     $i = i * 2$ 
```

Selected Answer:  [None Given]

Answers:

$\Theta(\log n)$

 $\Theta(n)$

$\Theta(n \log n)$


$\Theta(n^2)$

$\Theta(n^3)$

$\Theta(2^n)$

For følgende algoritme, hvad er den asymptotiske køretid i Θ -notation som funktion af n ?

```
ALGORITME3( $n$ )  
   $i = 1$   
  while  $i \leq n$   
     $j = 1$   
    while  $j \leq i$   
       $j = j + 1$   
     $i = i * 2$ 
```

Selected Answer:  [None Given]

Answers:

$\Theta(\log n)$

 $\Theta(n)$

$\Theta(n \log n)$

$\Theta(n^2)$

$\Theta(n^3)$

$\Theta(2^n)$

For følgende algoritme, hvad er den asymptotiske køretid i Θ -notation som funktion af n ?


```
ALGORITME4( $n$ )  
   $i = 1$   
  while  $i \leq n$   
     $j = 1$   
    while  $j \leq i$   
       $j = j * 2$   
     $i = i + 1$ 
```

Selected Answer:  [None Given]

Answers:

$\Theta(\log n)$

$\Theta(n)$

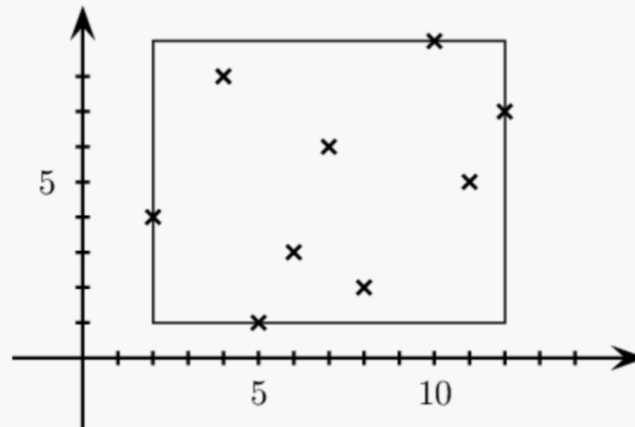
 $\Theta(n \log n)$

$\Theta(n^2)$

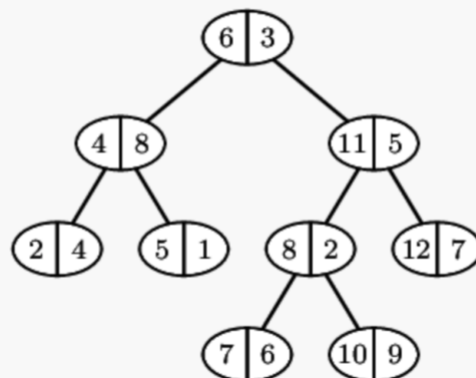
$\Theta(n^3)$

$\Theta(2^n)$

I denne og de næste opgaver ønsker vi at gemme en mængde punkter i planen, og at kunne svare på spørgsmål om det akse-parallelle rektangel, de udspænder. Et eksempel på en mængde punkter og deres udspændte rektangel er illustreret herunder.



Vi gemmer punkterne i knuderne i et søgetræ, hvor vi bruger punkternes x -koordinater som søgenøgle. Det må antages at ingen x -koordinat optræder i to forskellige punkter. Et eksempel på et sådant træ er vist herunder. Punkternes x -koordinater (træets søgenøgler) er angivet til venstre i knuderne, punkternes y -koordinater er angivet til højre.



Vi udstyrer nu enhver knude v i søgetræet med følgende fire ekstra informationer (udover de i v gemte x - og y -koordinater, som vi kalder $v.x$ og $v.y$):

$$\begin{aligned}v.xmax &= \text{største } x\text{-værdi i } v\text{'s undertræ} \\v.xmin &= \text{mindste } x\text{-værdi i } v\text{'s undertræ} \\v.ymax &= \text{største } y\text{-værdi i } v\text{'s undertræ} \\v.ymin &= \text{mindste } y\text{-værdi i } v\text{'s undertræ}\end{aligned}$$


(Husk at en knudes undertræ inkluderer knuden selv.)

Som eksempel vil det højre barn af roden i træet ovenfor have følgende informationer:

$$\begin{aligned}v.xmax &= 12 \\v.xmin &= 7 \\v.ymax &= 9 \\v.ymin &= 2\end{aligned}$$

For en knude v , hvordan kan dens informationer bestemmes i $O(1)$ tid ud fra informationerne i dens to børn $v.left$ and $v.right$?

(Et eller begge af børnene kan være NIL, hvilket giver simple specialtilfælde som vi ikke medtager her.)

Selected Answer:  [None Given]

Answers:


$$\begin{aligned}v.xmax &= v.left.xmax \\v.xmin &= v.right.xmin \\v.ymax &= \max(v.left.ymax, v.right.ymax, v.y) \\v.ymin &= \min(v.left.ymin, v.right.ymin, v.y)\end{aligned}$$

$v.xmax = v.right.xmax$
 $v.xmin = v.left.xmin$
 $v.ymax = \min(v.left.ymax, v.right.ymax, v.y)$
 $v.ymin = \max(v.left.ymin, v.right.ymin, v.y)$

$v.ymax = v.right.ymax$
 $v.ymin = v.left.ymin$
 $v.xmax = \max(v.left.xmax, v.right.xmax, v.x)$
 $v.xmin = \min(v.left.xmin, v.right.xmin, v.x)$

✓ $v.xmax = v.right.xmax$
 $v.xmin = v.left.xmin$
 $v.ymax = \max(v.left.ymax, v.right.ymax, v.y)$
 $v.ymin = \min(v.left.ymin, v.right.ymin, v.y)$

Vi lader nu søgetræet være et rød-sort træ. Hvilke af nedenstående svar er korrekte argumenter for, at informationerne i træets knuder kan vedligeholdes under indsættelser og sletninger, uden at rød-sort træers køretid på $O(\log n)$ for indsættelser og sletninger ændres?


Selected Answer:  [None Given]

Answers:

Ingen informationer i knuder skal ændres under indsættelser og sletninger, så de rød-sort træers $O(\log n)$ køretid gælder stadig.

Kun roden får ny information, og dette er $O(1)$ ekstra arbejde oven i de rød-sort træers $O(\log n)$ køretid, hvilket stadig er $O(\log n)$.

Ved indsættelser skal kun det nye blad have ny information, og dette er $O(1)$ ekstra arbejde oven i de rød-sort træers $O(\log n)$ køretid, hvilket stadig er $O(\log n)$. Ved sletninger kan information kun forsvinde, hvilket ikke giver anledning til arbejde.

-  Efter forandringer i træet under indsættelser og sletninger i de rød-sort træer skal kun knuder på en sti fra et blad til roden have ny information, og denne kan gendannes nedefra og op i tid $O(\log n)$.

Efter forandringer i træet under indsættelser og sletninger i de rød-sort træer kan et inorder gennemløb af træet genoprette informationen i alle knuder i tid $O(\log n)$.

3 point

Hvordan kan arealet af det rektangel, som de gemte punkter udspænder, bestemmes i $O(1)$ tid ud fra træet?

Selected Answer:  [None Given]

Answers:

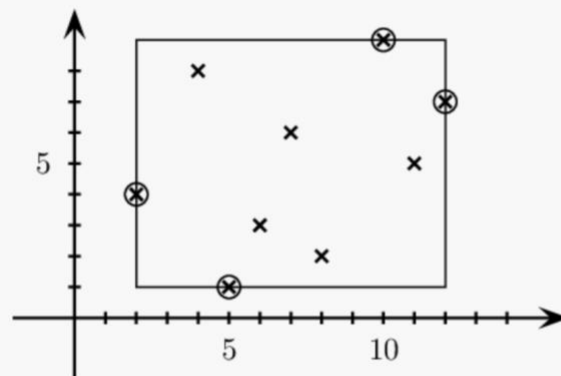
Dette areal kan findes som $v.ymin \times u.ymax$, hvor v er bladet mest til venstre i træet, og u er bladet mest til højre i træet.

Dette areal kan findes som $r.xmax \times r.ymax$, hvor r er roden i træet.


✓ Dette areal kan findes som $(r.xmax - r.xmin) \times (r.ymax - r.ymin)$, hvor r er roden i træet.

Dette areal kan findes ved et inorder gennemløb af træet, hvor man finder de mindste og de største x - og y -værdier. Derudfra kan man beregne sidelængderne af det udspændte rektangel, og derfra arealet.

Vi ønsker nu for hver af de fire sider af det udspændte rektangel at finde et punkt som ligger på denne side. Som illustration ønsker vi i eksemplet fra før at finde de fire punkter fremhævet nedenfor.



Hvordan kan man gøre dette i $O(\log n)$ tid ved hjælp af træet? [Et eller flere svar.]

Selected Answers:  [None Given]

Answers:

Punkterne på de vandrette sider kan findes med TREE-MAXIMUM og TREE-MINIMUM fra lærebogen. Punktet på den højre, lodrette side kan findes ved en rekursiv søgealgoritme, som starter i roden, og som via $xmax$ -værdierne bevæger sig mod knuden med den største x -koordinat. Punktet på den venstre, lodrette side kan findes på samme måde, med $xmin$ i stedet for $xmax$.

✓ Punkterne på de lodrette sider kan findes med TREE-MAXIMUM og TREE-MINIMUM fra lærebogen. Punktet på den øverste, vandrette side kan findes ved en rekursiv søgealgoritme, som starter i roden, og som via $ymax$ -værdierne bevæger sig mod knuden med den største y -koordinat. Punktet på den nederste, vandrette side kan findes på samme måde, med $ymin$ i stedet for $ymax$.

Punkterne på både de vandrette og lodrette sider kan findes med TREE-MAXIMUM og TREE-MINIMUM fra lærebogen.

✓ Punktet på den øverste, vandrette side kan findes ved en rekursiv søgealgoritme, som starter i roden, og som via $ymax$ -værdierne bevæger sig mod knuden med den største y -koordinat. Punktet på den nederste, vandrette side kan findes på samme måde, med $ymin$ i stedet for $ymax$. Punktet på den højre, lodrette side kan findes ved en rekursiv søgealgoritme, som starter i roden, og som via $xmax$ -værdierne bevæger sig mod knuden med den største x -koordinat. Punktet på den venstre, lodrette side kan findes på samme måde, med $xmin$ i stedet for $xmax$.

Punkterne på alle fire sider kan findes ved et inorder gennemløb af træet.

Vi vil nu gerne opnå ovenstående resultater—nemlig at kunne finde arealet af det udspændte rektangel i $O(1)$ tid og at kunne finde fire punkter, et på hver side af rektanglet, i $O(\log n)$ tid—på en anden måde. Indsættelser og sletninger skal stadig kunne foretages i $O(\log n)$ tid.

Det må antages at ingen x -koordinat optræder i to forskellige punkter og at heller ingen y -koordinat optræder i to forskellige punkter.


Det påstås, at dette kan opnås ved at bruge to træer uden ekstra information i alle knuder. Hvilke af følgende datastrukturer tillader dette?

Selected Answer:  [None Given]

Answers:

Det ene træ er en max-heap, hvor punkternes x -koordinater er nøgler. Det andet træ er en max-heap, hvor punkternes y -koordinater er nøgler.

Det ene træ er en max-heap, hvor punkternes x -koordinater er nøgler. Det andet træ er et rød-sort træ, hvor punkternes y -koordinater er nøgler, og hvor man gemmer største og mindste y -koordinat i træet i to globale variable.

 Det ene træ er et rød-sort træ hvor punkternes x -koordinater er nøgler, og hvor man gemmer største og mindste x -koordinat i træet i to globale variable. Det andet træ er et rød-sort træ hvor punkternes y -koordinater er nøgler, og hvor man gemmer største og mindste y -koordinat i træet i to globale variable.

Det ene træ er et Huffman-træ hvor punkternes x -koordinater er frekvenser. Det andet træ er et Huffman-træ hvor punkternes y -koordinater er frekvenser.