Written exam in ADA5

Algorithms and data structures

January 6th, 2022

Dansk version på siderne 5-8

There is a total of 6 exercises/problems for this exam.

Hand in one and only one pdf-document. Code that you wish to be tested must be editable and therefore *not* included as screenshots.

For the programming exercises you are free to choose between Java and C++.

You may write your explanations in Danish.

Problem 1 (10 %)

What is the Big-Oh time complexity of the following method? Please explain your answer.

```
public static int myMethod( int N )
{
   int x = 0; int y = 0;
   for (int i = 0; i < N; i++)
   {
        for (int j = 0; j < N; j++)
        {
            for (int k = 0; k < N*Math.sqrt(N); k++)
             //square root; C++: #include <math.h>
            {
                x++;
            }
            j *= 2;
        }
        i += i;
   }
   for (int i = 0; i < N*N; i++)
        y++;
   return x+y;
}
```

Problem 2 (15 %)

Write a *recursive* method/algorithm with the following signature:

```
bool additive(String s)
```

The parameter contains a string of numbers, e.g. “82842605”.

The algorithm returns true if the string contains a substring of three consecutive numbers where the third character is the sum of the two previous characters.

In the example above the method returns true, because index 5 (6) is the sum of indices 3 and 4 (4 plus 2).

Hint: the ASCII value of the character ‘7’ is 55.

Problem 3 (30 %)

Write an algorithm that takes an array of unsorted, unique natural numbers as input and finds the three numbers in the array whose sum is closest to a power of 2. The same number can only be used once.

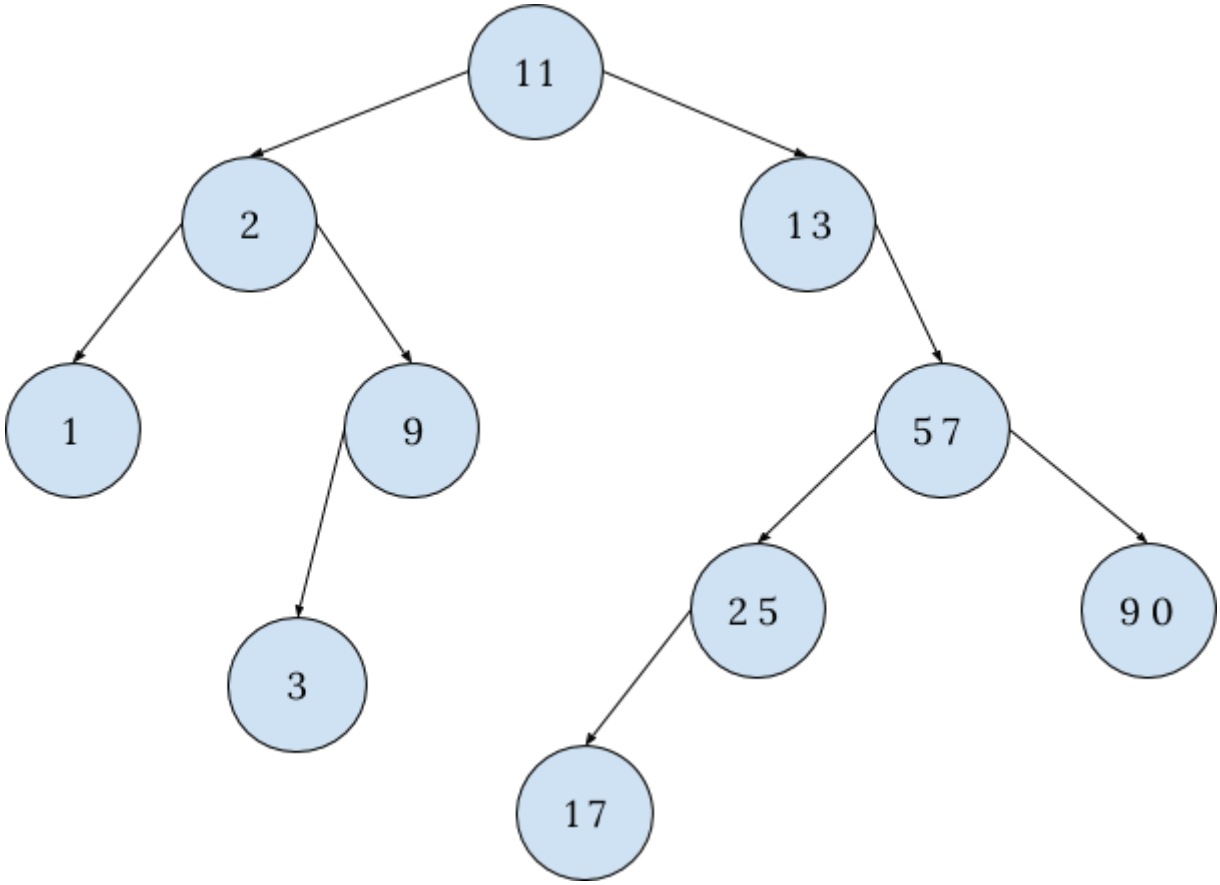
The return value of your algorithm must be an integer array that contains first the three values and then the corresponding power of two (e.g. 512).

Called with the array {23,56,22,11,65,89,3,44,87,910,45,35,98}, the three numbers returned are 89, 3, 35 and the power of 2 is 128.

What is the Big-Oh time complexity of your algorithm? Explain your answer and discuss the possibilities of optimizing your solution.

Problem 4 (17 %)

The figure below shows a binary search tree, which is not an AVL-tree:



List the order in which nodes will be visited in an in-order and in a level-order traversal.

What is the internal path length of the tree?

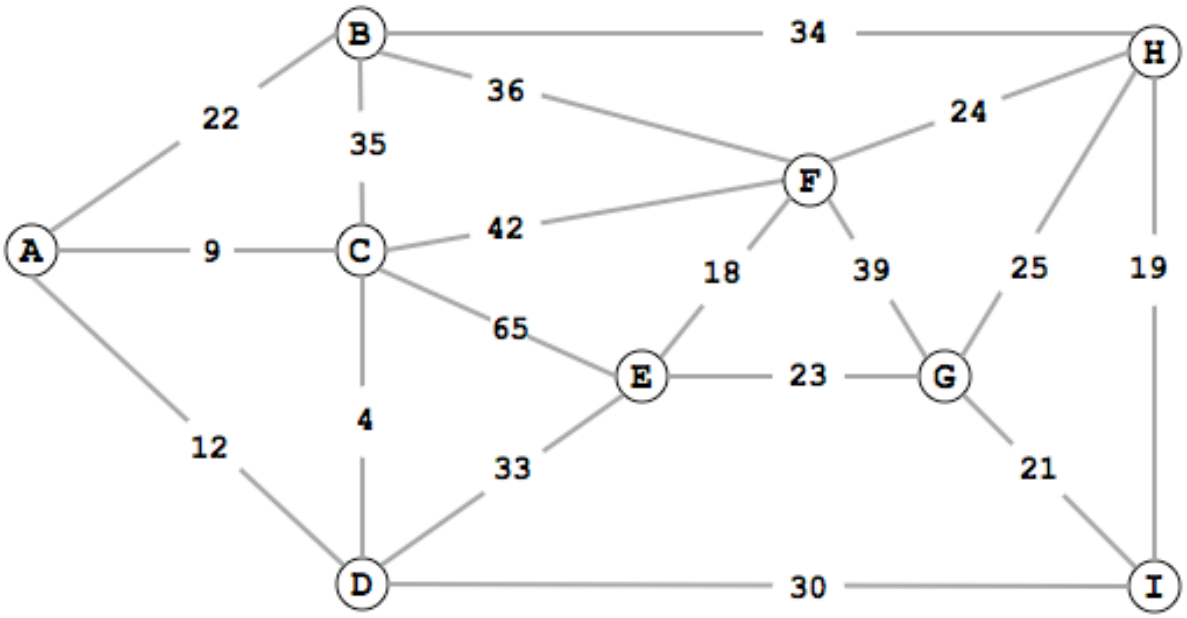
The imbalance of the tree is at node 13 where the height of the right subtree is 3 and the height of the left subtree is 0.

How could you rearrange the nodes of the root's right subtree, so that it becomes an AVL-tree?

Could the tree have been an AVL-tree before the previous updating operation (insert or delete, but not rotation)? Examples of possible previous updating operation could be inserting node 3 or deleting node 12 (left child of node 13). Explain your answer.

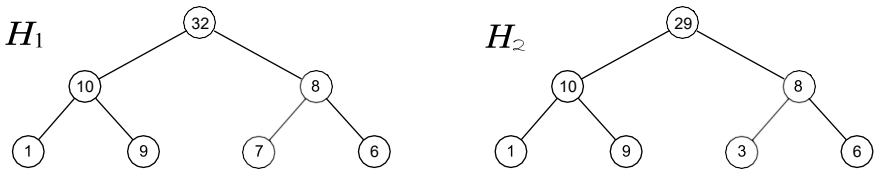
Problem 5 (18 %)

Find a *minimum spanning tree* for graph below using Prim's algorithm. Your answer must be a list of edges, E-G. D-E, A-C etc., which shows in which order the algorithm will establish connections between vertices.



Could you end up with a different answer if Kruskal's algorithm were applied? Explain your answer.

Problem 6 (10 %)



H_1 and H_2 are max heaps. A max heap is a priority queue where the biggest key is in the root, and the definition of the heap order is that any node must have a key that is bigger than all its descendants'.

Draw H_1 after an *insert* operation with key 12.

Draw H_2 after a *deleteMax* operation.

Skriftlig eksamen i ADA5

Algoritmer og datastrukturer

6. januar, 2022

Der er i alt 6 opgaver til denne eksamen.

Der afleveres ét og kun ét pdf-dokument. Kode, som du ønsker testet, skal være editérbar og derfor *ikke* i form af screenshots.

I programmeringsopgaverne kan du vælge frit mellem Java og C++.

Du må gerne skrive dine forklaringer på dansk.

Opgave 1 (10 %)

Hvad er Store O (Big Oh) tidskompleksiteten af nedenstående metode. Begrund dit svar.

```
public static int myMethod( int N )
{
   int x = 0; int y = 0;
   for (int i = 0; i < N; i++)
   {
        for (int j = 0; j < N; j++)
        {
            for (int k = 0; k < N*Math.sqrt(N); k++)
            //square root; C++: #include <math.h>
            {
                x++;
            }
            j *= 2;
        }
        i += i;
   }
   for (int i = 0; i < N*N; i++)
        y++;
   return x+y;
}
```

Opgave 2 (15 %)

Skriv en *rekursiv* metode/algoritme med følgende signatur:

```
bool additive(String s)
```

Parameteren indeholder en tekststreng af cifre, fx “82842605”.

Algoritmen returnerer true, hvis tekststrengen indeholder en substring bestående af tre på hinanden følgende cifre, hvorom det gælder, at det tredje ciffer er lig med summen af de to foregående cifre.

I ovenstående eksempel returnerer metoden true, fordi indeks 5 (6) er lig med summen af indeks 3 og 4 (4 plus 2).

Tip: ASCII-værdien af karakteren ‘7’ er 55.

Opgave 3 (30 %)

Skriv en algoritme, som tager et array af usorterede, unikke naturlige tal som input og finder de tre tal i arrayet, hvis sum er tættest på en potens af 2. Det samme tal kan kun forekomme én gang.

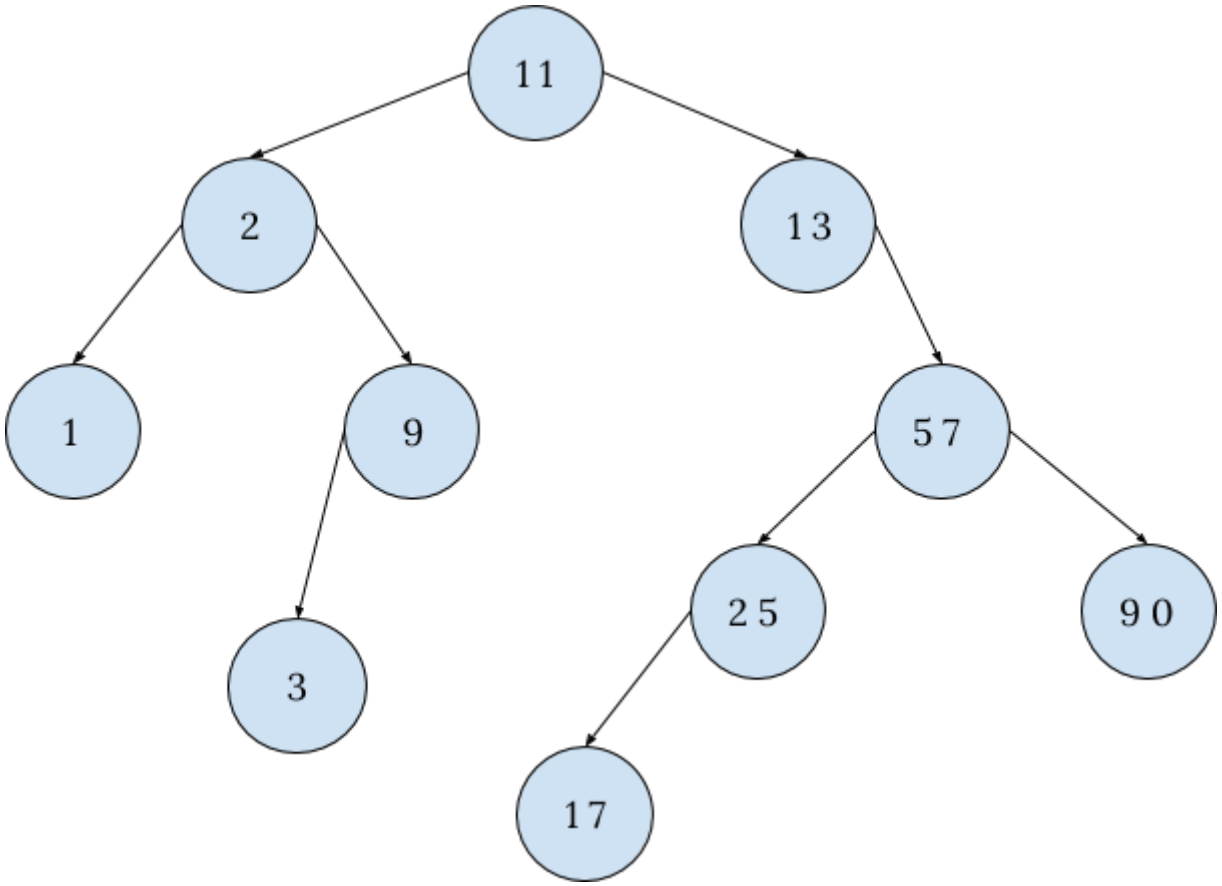
Din algoritmes returværdi skal være et integer array, som først indeholder de tre tal og dernæst potensen af 2 (fx 512).

Kaldt med arrayet {23,56,22,11,65,89,3,44,87,910,45,35,98}, returneres de tre tal 89, 3, 35, og potensen af to er 128.

Hvad er Store O (Big Oh) tidskompleksiteten af din algoritme? Diskutér mulighederne for optimering af din løsning.

Opgave 4 (17 %)

Nedenstående figur viser et binært søgetræ, som ikke er et AVL-træ:



List den rækkefølge, som noderne bliver besøgt i i en in-order og i en level-order traversering (traversal).

Hvad er træets internal path length?

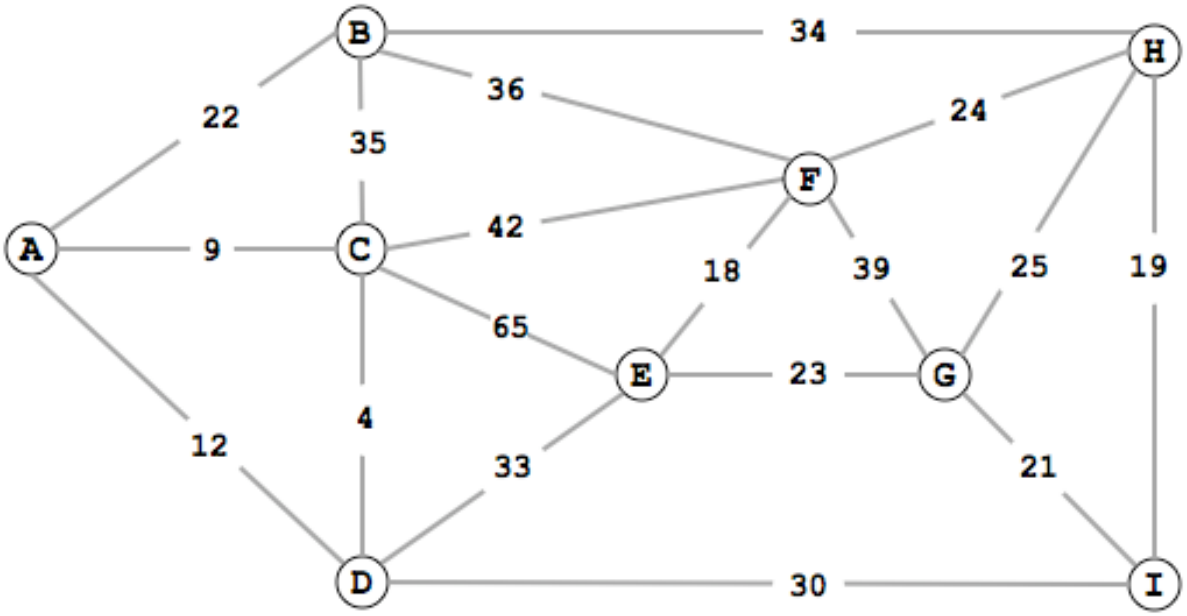
Træets ubalance findes i node 13, hvor højden af det højre subtræ er 3, og højden af det venstre subtræ er 0.

Hvordan kan noderne I rodens højre subtræ reorganiseres, således at træet bliver et AVL-træ?

Kunne træet have været et AVL-træ før den seneste opdaterende operation (insert eller delete, men ikke rotation)? Eksempler på seneste opdaterende operation kunne være indsættelsen af node 3 eller sletning af node 12 (node 13's venstre barn). Begrund dit svar.

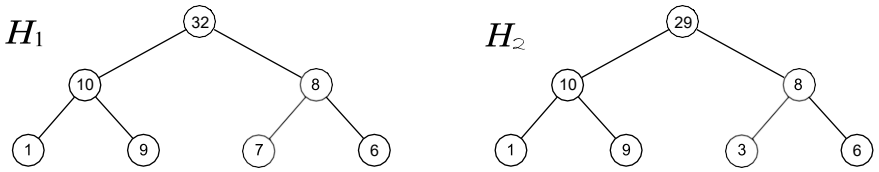
Opgave 5 (18 %)

Find et *minimum spanning tree* for nedenstående graf ved anvendelse af Prims algoritme. Dit svar skal være en liste af kanter (edges), som viser i hvilen rækkefølge, algoritmen vil etablere forbindelser mellem noderne (vertices).



Kunne løsningen være en anden, hvis Kruskals algoritme var blevet anvendt? Begrund dit svar.

Opgave 6 (10 %)



H_1 og H_2 er max heaps. En max heap er en prioritetskø, hvor den største nøgle er i roden, og definitionen af heap order tilsiger, at en hvilken som helst node skal have en nøgle, som er større end alle dens efterkommeres nøgle.

Tegn H_1 efter en *insert* operation med nøglen 12.

Tegn H_2 efter en *deleteMax* operation.