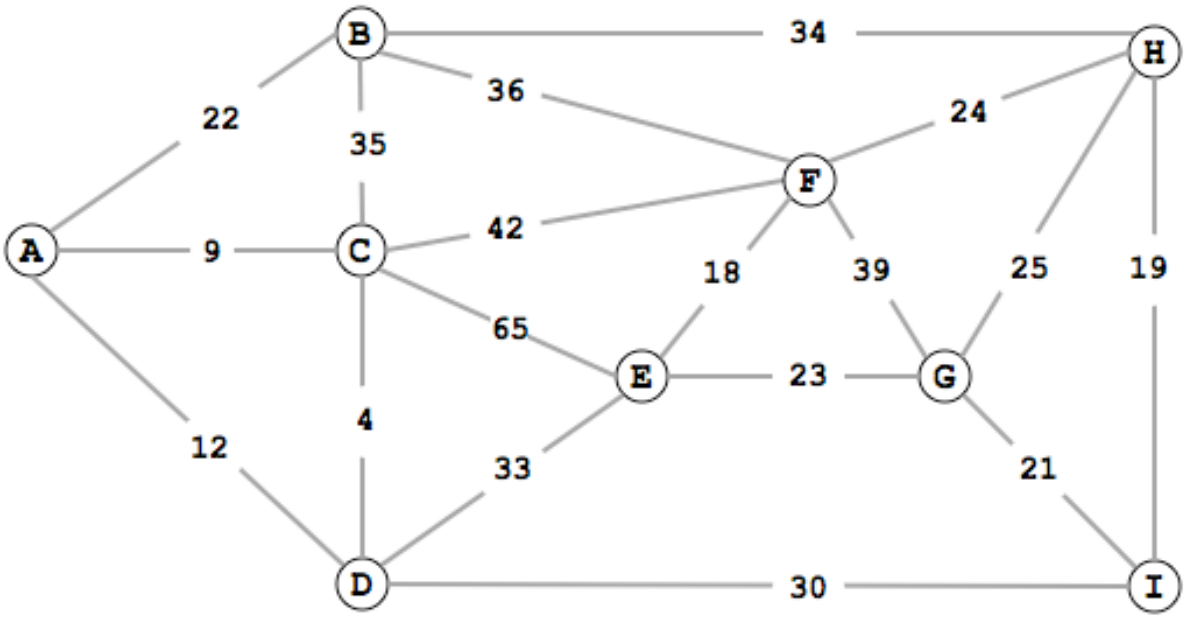
Exercise 1

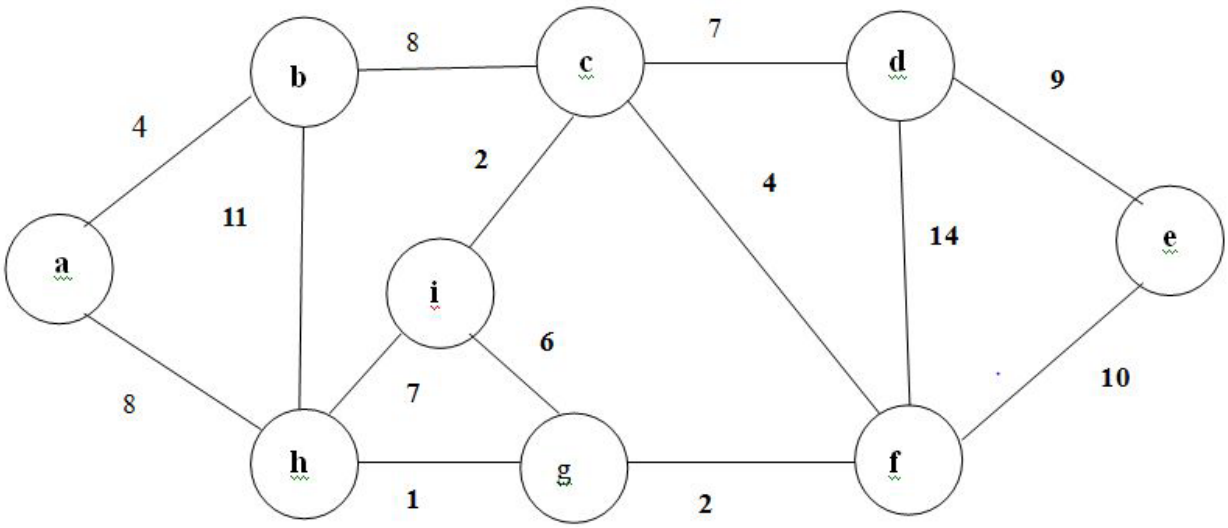
Find a *minimum spanning tree* for graph below using Prim's algorithm. Your answer must be a list of edges, E-G. D-E, A-C etc., which shows in which order the algorithm will establish connections between vertices.



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

Exercise 2

Find a *minimum spanning tree* for below graph using Kruskal'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.



Exercise 3

Write a recursive method with the following signature:

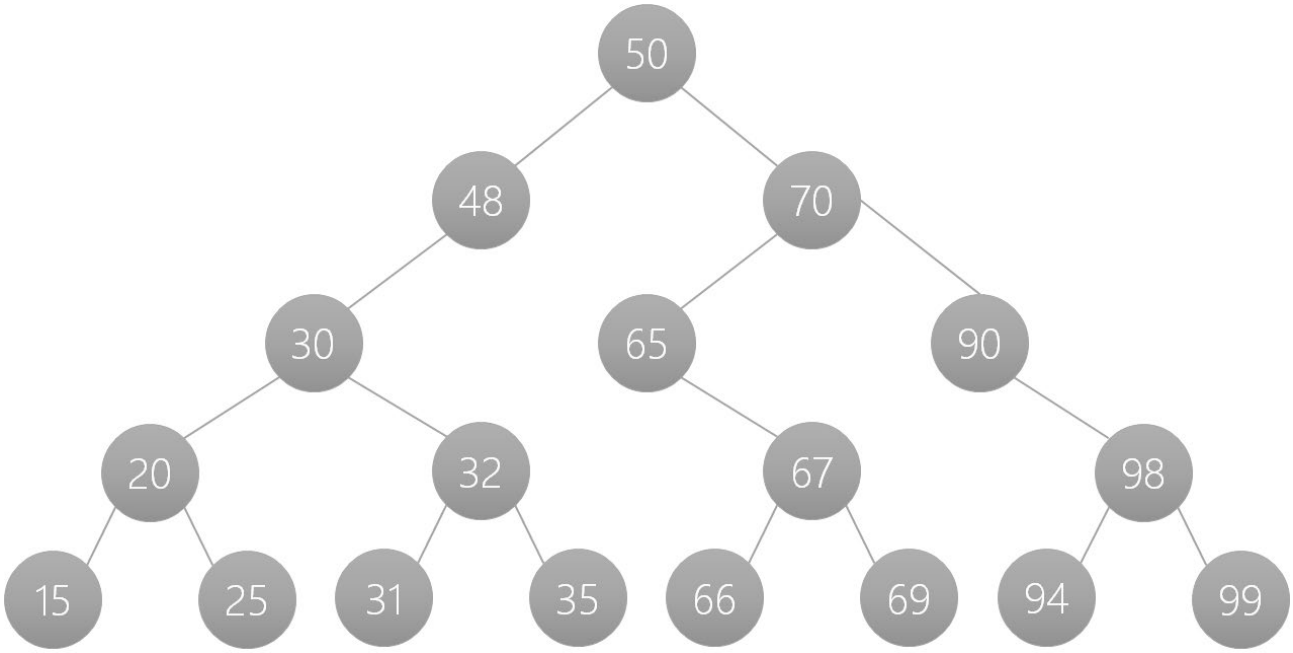
```
int logTwo(int N)
```

The algorithm returns the base 2 algorithm of N, and it is a precondition that N is a positive integer and a power of 2.

Called with `N = 32` it returns 5, og with `N = 4096` it returns 12.

Exercise 4

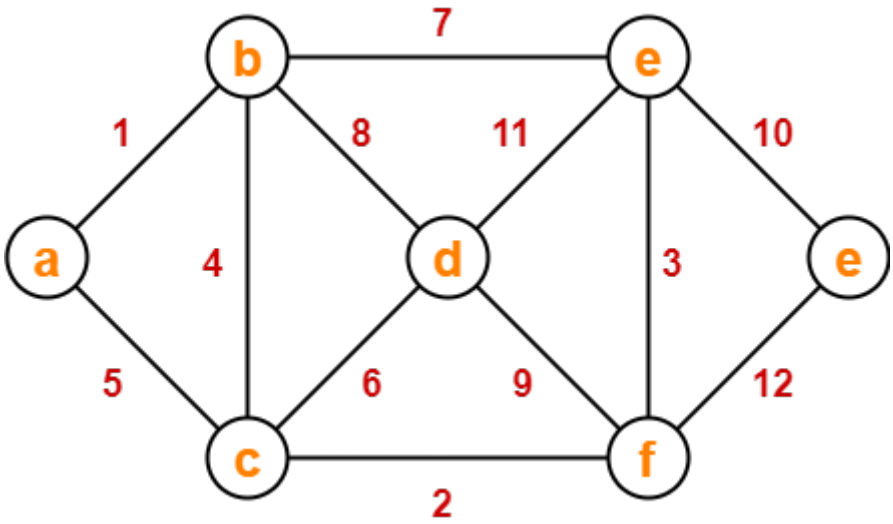
The figure below is a binary search tree:



List the order in which the nodes will be visited in an in-order and a level order traversal.
Which simple data structure can preferably be used for a level order traversal and why?
What is the internal path length of the tree, and what other characteristics does the tree have? Explain your answers.

Exercise 5

Find a *minimum spanning tree* for the graph below using Prim's algorithm. Your answer must be a list of edges, fx d-e, a-c etc., which show the order in which the algorithm will establish connections between the nodes (vertices). Also give the size of the tree (the sum of the weights in the tree).



Exercise 6

The table below is a hash table. The hash function is X % tableSize, and the unique key af the element is the number of the letter in the alphabet. Thus the letter E has the key 5, and the letter O has the key 15. For collision resolution *quadratic probing* is used.

Index	Value
0	
1	A
2	W
3	C
4	O
5	E
6	
7	
8	S
9	
10	

The load factor of the table is above 0.5 and a rehashing should be performed. Show the table after the rehashing has been performed.

Exercise 7

The following iterative sequence is defined for the set of positive integers:

$n \rightarrow n/2$ (n is even)
$n \rightarrow 3n + 1$ (n is odd)

Using the rule above and starting with 13, we generate the following sequence:

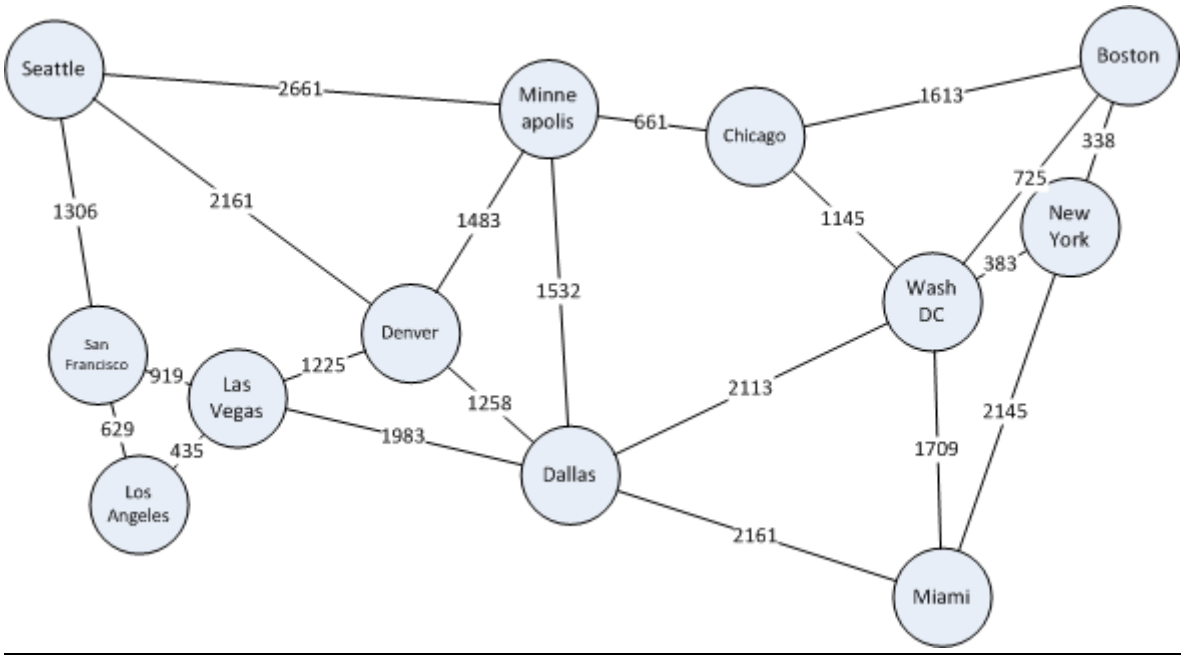
$$13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

This sequence (starting at 13 and finishing at 1) contains 10 terms, and it may be assumed that all starting numbers finish at 1.

Write an algorithm that can determine which starting number, under 10,000 (ten thousand), that produces the longest chain.

What can be said about the running time of your algorithm, and can it possibly be improved?

Exercise 8



Apply Dijkstra's shortest path algorithm to above graph with Seattle as starting point.

The answer must give a short explanation of how the algorithm operates and a list of used edges naming the two vertices (nodes) involved, for example

Minneapolis-Dallas
Chicago-Wash DC
New York-Boston
etc.

Exercise 9

A Pythagorean triplet is a set of three natural numbers, $a < b < c$, for which,

$$a^2 + b^2 = c^2$$

For example, $3^2 + 4^2 = 9 + 16 = 25 = 5^2$.

There exists exactly one Pythagorean triplet for which $a + b + c = 1000$.

Write an algorithm that can print out the values of a, b and c fulfilling the requirement (a+b+c=1000) and return the product *a*b*c*.

What can be said about the running time of your algorithm, and can it possibly be improved?

HINT: The correct return value is 31.875.000.

Exercise 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;
   for (int i = 0; i < N; i++)
        for (int j = 0; j < N/2; j++)
            for (int k = 1; k < N;)
            {
                x++;
                k *= 2;
            }
   return x;
}
```

Exercise 11

Write a recursive method/algorithm with the following signature:

```
int getNumberEqual(int[] x, int n, int val);
```

The algorithm returns the number of occurrences of a certain integer (parameter `val`) in an integer array (parameter `x`) Parameter `n` is the number of elements in the array.

Called with the array {7,4,1,3,5,6,4,8} , `n`=8, and `val`=4, the correct return value is 2.

Exercise 12

Solve the following recurrence. All relevant steps must be shown leading to a Big-Oh answer.

$$T(N) = T(N-1) + (N-1)$$

Exercise 13

The task is to specify a method that can change the priority of an element in a priority queue of integers and reestablish the heap order. The signature could be

```
changePriority(int fromP, int toP);
```

Preconditions of the method are that an element with priority `fromP` exists, and that `toP != fromP`.

Explain in detail, step by step, how you would write/implement the method (in ordinary language/pseudo code, you do not need to implement it in a real programming language)

Exercise 14

Consider the hash-table below

Index	Value
0	D
1	H
2	V
3	
4	
5	P
6	
7	X
8	E
9	
10	

Indexes 3, 4, 6, 9 and 10 are vacant. What happens if an element hashing to index 1 is to be inserted? The strategy used for collision resolution is quadratic probing.

Exercise 15

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

```
public static int myMethod1( int[] arr )
{
            int x = 0;
            for (int i = 0; i < arr.length/2; i++)
                        for (int j = 0; j < arr.length; j++)
                                    for (int k = 0; k < arr.length; k++)
                                    {
                                                x++;
                                                if (k==arr.length/2)
                                                            break;
                                    }
            return x;
}
```

The attribute *length* denotes the number of elements in the array.

Exercise 16

An array contains integers between 0 and 100 with duplicates. Explain how you could create an algorithm (you do not need to code it) that can return the most frequently occurring integer in the array. The algorithm must have O(N) time complexity.
Example: on the array {5,28,7,25,7,9,28,11,67,5,33,28} the algorithm will return 28.

Exercise 17

Write a recursive method/algorithm that takes a positive integer N as a parameter and returns the sum of the integers from 1 to N.

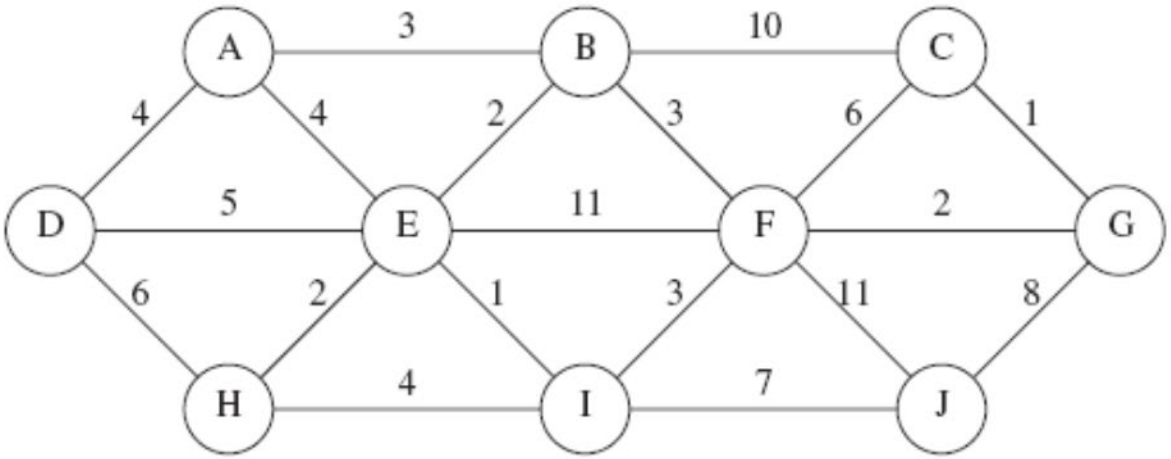
Example: called with the parameter 5 the method returns 15 (1+2+3+4+5).

Exercise 18

Solve the following recurrence. All relevant steps must be shown leading to a Big-Oh answer.

$$T(N) = T(N/2) + N$$

Exercise 19



Find a minimum spanning tree of the above graph using Prim's algorithm. Your answer must be a list of edges, e.g. E-I, F-J etc., showing the order in which the algorithm will establish connections between vertices.

Exercise 20

The following rules apply to the password to some system:

- The password always contains six digits [1-9] (zero is not a valid digit).
- Digits must appear in ascending order (123567 is legal; 124563 is not).
- The same digit may only occur once.

Write an algorithm that can calculate the number of possible passwords for this system and state the number.

In Java you can convert an integer to a string using: String str = Integer.toString(i);

In C++ you can convert an integer to a string using: string str= to_string(i);

Exercise 21

Write a recursive method that has a positive integer as parameter and returns the sum of the even numbers squares from 1 to N.

Example: called with the parameter 9, the method returns 120 ($2^2 + 4^2 + 6^2 + 8^2$).

Exercise 22

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

```
public static int myMethod( int[] arr )
{
      int x = 0;
      for (int i = 0; i < arr.length; i++)
           for (int j = 0; j < arr.length/2; j++)
               for (int k = 0; k < arr.length; k++)
               {
                   x++;
                   if (k==1)
                       break;
               }
      return x;
}
```

The attribute *length* denotes the number of elements in the array.

Exercise 23

Write a recursive method/algorithm that takes a positive integer N as a parameter and returns the sum of the odd integers from 1 to N.

Example: called with the parameter 9 the method returns 25 (1+3+5+7+9).

Exercise 24

Write a method to the *BinarySearchTree* data structure that can decide if the tree is of minimal height, e.g.

```
bool minimalHeight();
```

Examples: if the tree contains 27 nodes, the minimal height is 4; if the tree contains 12 nodes, the minimal height is 3.
Hint: if the number of nodes is 27 the minimal height can be found by calculating the next power of 2 which is 32 (= 2^5) and subtracting one from the result. The minimal height will then be five minus one.

C++ only:
To determine the height of the tree you can use the following *public* and *private* methods (the methods are available in Weiss' java-version of the binary search tree).

Public:

```
int BinarySearchTree::height()
{
            return height(root);
}
```

Private/internal:

```
int BinarySearchTree::height(BinaryNode *t)
{
            if (t == nullptr)
                        return -1;
            else
                        return 1 + max(height(t->left), height(t->right));
}
```

Exercise 25

Two words are anagrams if they contain the same letters with the same frequency. *inch* and *chin* are anagrams of each other, and so are *state* and *taste.*

What would be the best Big-Oh runtime of an algorithm that can solve this problem? Explain your answer (you do not need to write the code).

Suppose that the two words to be checked are randomly generated, for instance by lookups in a dictionary.

How could your algorithm's runtime then be significantly improved, and what additional information would you need to estimate the improved algorithm's time complexity?