Written exam

BTH004 – The design and analysis of algorithms

---

1. This question contains 10 subquestions (a-j) where you are expected to provide short answers, which you do not need to motivate (most of the questions are yes/no questions). For each correct answer you get 2p and for each incorrect answer you get -1p. The maximum score for this question, if you provide the correct answer to all subquestions, is 20p. The minimum score for this question is 0p; even if you have a negative total score on this question you will get 0 points.

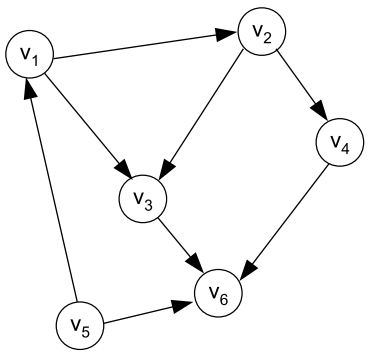
    a. Is the function $f(n) = n^7 + 20n^2$ polynomially bounded?
    Answer: Yes

    b. Is it correct to say that neighborhood search is a type of improving search heuristics?
    Answer: Yes

    c. Is it correct to say that Prim's algorithm is based on the principles of dynamic programming?
    Answer: No

    d. What are the Bellman equations used for in the shortest path problem?
    Answer: They define the optimality conditions

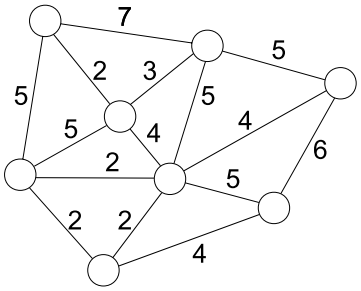
    e. Does the sequence $v_1 \rightarrow v_5 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow v_6$ of vertices define a correct topological ordering for the following directed graph?



Answer: No

f. Is the word *nation* a suffix of the word *examination.*
Answer: Yes

g. Are there multiple optimal solutions in the following instance of the minimum spanning tree problem?



Answer: Yes (there is "cycle" containing edges with value 2)

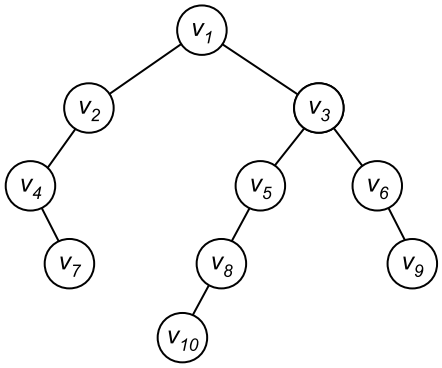
h. Is it true that the function $f(n) = 2n^2 + 4$ is $O(n^3)$?
Answer: Yes

i. What is the basic operation in sorting?
Answer: Comparing elements

j. What is a constructive heuristic?
Answer: A method that iteratively "builds up" a solution to an optimization problem.

2. This question tests your knowledge on search trees and graphs.

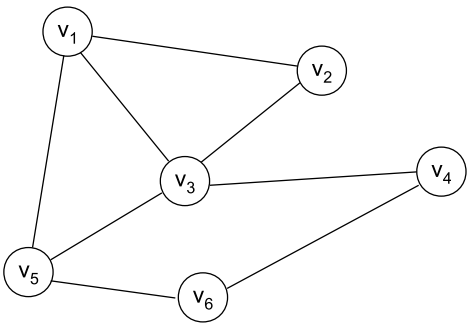
a. Consider the following binary search tree, which is rooted in node v_1.



Order the nodes from smallest to largest based on their locations in the tree (6p)
Answer: $v_4 < v_7 < v_2 < v_1 < v_{10} < v_8 < v_5 < v_3 < v_6 < v_9$

b. What is a balanced binary tree? (2p)
Answer : A tree where the height difference of the left and the right subtrees of each of the vertices/nodes in the tree differs with at most 1.

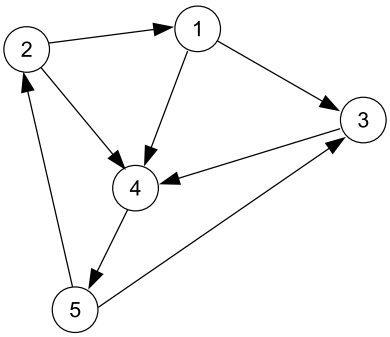
c. For the following undirected graph, provide an equivalent directed graph. (3p)



Answer: A graph with the following directed arcs: (v1->v2), (v2->v1), (v1->v3), (v3->v1), (v1->v5), (v5-v1), (v2->v3), (v3->v2), (v3->v4), (v4->v3), (v3->v5), (v5->v3), (v4->v6), (v6->v4), (v5->v6), (v6->v5).

d. Construct the least height red-black tree containing 7 nodes with values 1, 2, 3, 4, 5, 6, and 7. (4p)
Answer: The tree is a complete tree with only black nodes. The root contains the value 4. The children of the root, from left to right , are the values 2 and 6. The grandchildren of the root, from left to right, are the values 1, 3, 5, and 7

e. Create an adjacency matrix and an adjacency list for the following graph. Which of the graph representations is best from a complexity perspective for the given graph? (5p)



Answer:

Adjacency matrix:
(0 0 1 1 0
1 0 0 1 0

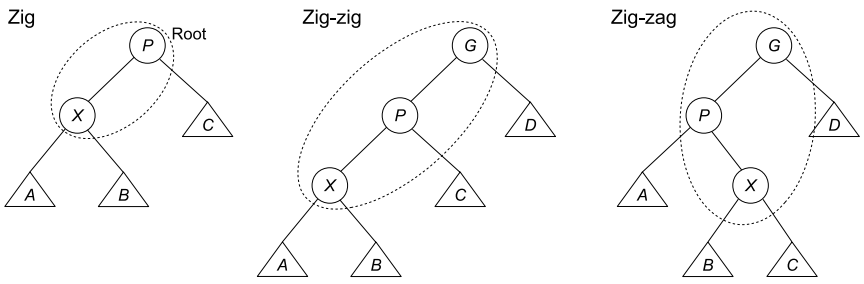
0 0 0 1 0
0 0 0 0 1
0 1 1 0 0)

Adjacency list:
1: 3, 4
2: 1, 4
3: 4
4: 5
5: 2, 3
Adjacency list is best from the complexity perspective.

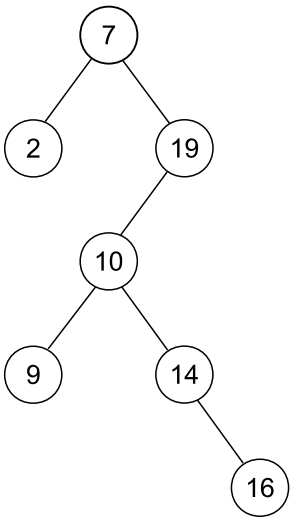
3. A splay tree is a type of self-adjusting binary search tree, which has an amortized cost of $O(log_2 n)$ per operation.

   Splay trees make use of splaying, which is similar to tree rotation, when inserting, deleting, and accessing nodes. Splaying moves the accessed node to the root of the tree.
   There are three types of splaying operations, which are performed bottom up along the access path of a node:

   zig - A single AVL rotation.
   zig-zig - Two AVL rotations in the same direction.
   zig-zag - A standard AVL double rotation.

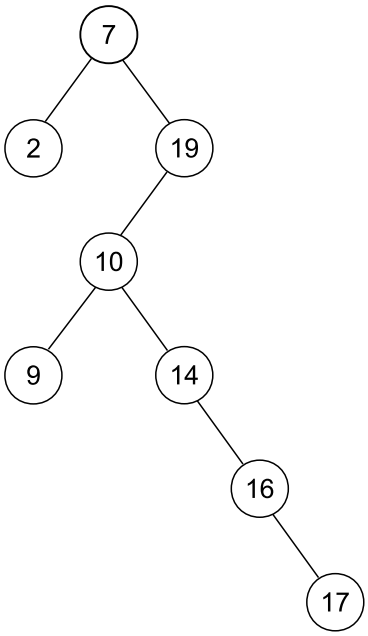


   Your task is to insert a node with value 17 in the tree below.

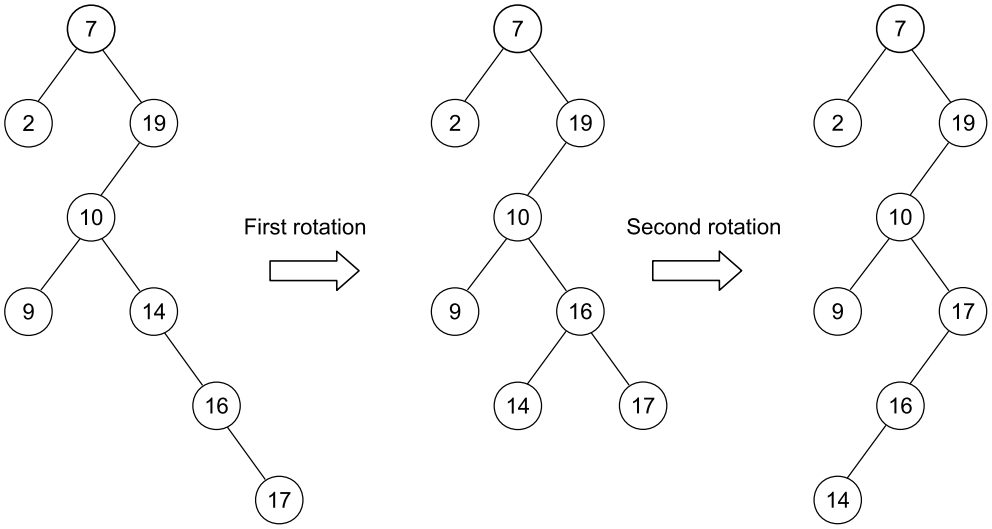


You are required to show what the tree looks like after inserting the new node and after each of the (3) splay operations. (20p)

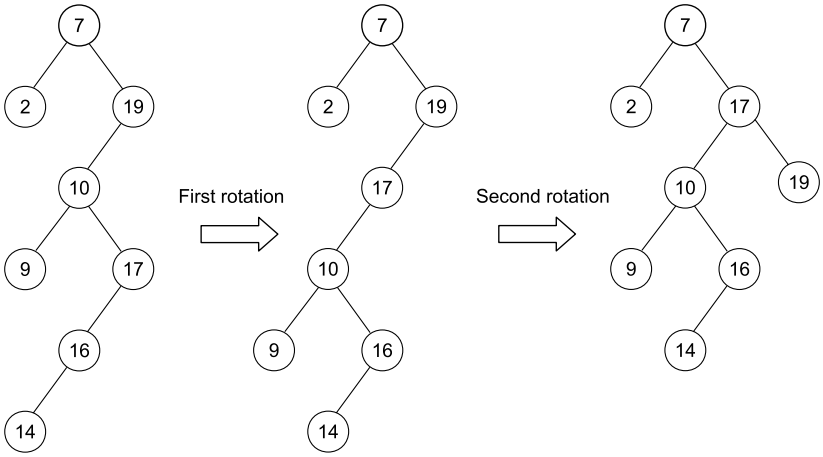
Answer: After insert the tree looks like this.



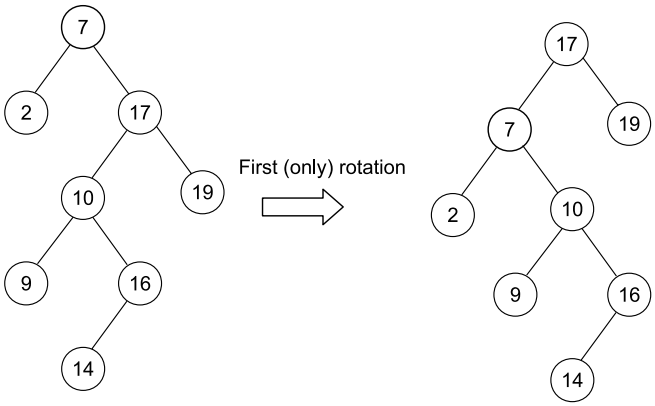
After the first (zig-zig) splay the tree looks like this:



After the second (zig-zag splay)



After the third (zig) splay



4. This question tests your knowledge on algorithms and algorithm construction.
   a. What is the problem with the following recursive function?

      function $f(x)$
        if x == 0 **then**
          return 0;
        else
          return $2 + f(x + 1)$

      (3p)

      Answer: It does not converge towards the base case if the function is called for values larger than 0.

   b. Mention two of the reasons, which were discussed in the course, for using a heuristics instead of an optimizing algorithm in order to solve an optimization problem. (3p)

      Answer:

      The following three reasons where discussed in the course.

      1. An optimizing algorithm might consume too much resources (memory and CPU).
      2. Input to the algorithm might be inaccurate and a heuristically good solution might be good enough.
      3. For a non-expert, it might be easier to understand a heuristic than an optimizing algorithm.

   c. Why it is required to provide a feasible starting solution when solving a problem using neighborhood search? (3p)

      Answer: Neighborhood search is an improving search algorithm that operates by iteratively modifying a feasible solution. If no feasible solution is provided, the algorithm does not have any feasible solution to modify.

   d. Why is it necessary to have at least one base case in a recursive function? (3p)

      Answer: Without a base case the recursive function will never terminate.

   e. Apply Huffman's algorithm in order to find an optimal code for the word *Mississippi*. (8p)

      Answer:

      Initialization: Create one tree with one node for each of the letters in the word. The trees have the following weights.

      M: 1
      i: 4
      s: 4
      p: 2

      Iteration 1: Merge the trees M and p into a new tree rooted in a new node

T1 where M and p are child nodes. The weight of the new tree is 3.

Iteration 2: Merge the trees T1 and i into a new tree rooted in a new node T2, which has the nodes T1 and i as children. The weight of the new tree is 7.

Iteration 3: Merge the trees T2 and s into a new tree rooted in a new node T3, which has the nodes T2 and s as children. The weight of the new tree is 11.

Now assign the 0 on the left going nodes and 1 on the right going links in the obtained tree.

This gives, for example, the code:
s = 0
i = 10
M = 110
p = 111

5. The question concerns complexity analysis.

   a. Describe the steps that are typically used to show that a problem P_1 is NP complete. (5p)
   Answer:
   1. Show that P1 is in NP
   2. Identify another problem P2, which is known to be in NP, and show that P1 can be polynomially reduced to P2.

   b. Describe why the complexity of an algorithm is usually described as a function $f(n)$ of the input size n, and provide an example illustrating your reasoning. (5p)
   Answer: The reason is that the input size (e.g., the number of elements to sort) typically has an important influence (sometimes the most significant influence) on the number of steps required by an algorithm. For example, a sorting algorithm with complexity $\Theta(n)^2$ will require approximately 100 basic operations to sort 10 algoritms and approximately 10000 basic operations to sort 100 elements.

   c. Explain how it can be easily shown that $P \subseteq NP$ (4p)
   Answer: A solution to all problems that can be solved in polynomial time can also be verified in polynomial time.

d. What is the worst-case complexity and average-case complexity of an algorithm? Why is there a need for both worst-case and average-case complexity analysis? (6p)

Answer:

The worst-case complexity W (n) of an algorithm is the maximum number of basic operations performed for any input of size n.

The average-case complexity A(n) of an algorithm is the average number of basic operations performed by the algorithm, where each input I occurs with some probability p(I).

The reason that both average case and worst case complexity analysis is needed is that some algorithms will perform different amount of work for different input.

Good luck!