# Data structures and algorithms
# (2019/20)
# Written exam 13. svečana 2020

This written exam must be taken individually. Any and all literature may be used while taking this test. In your answers be precise, and: (i) answer the questions *as they were asked*; and (ii) answer *all* tasks – if you will be answering to all tasks you might get bonus points.

Time: 90 minutes.

We wish you a lot of success - veliko uspeha!

| TASK | POINTS | OF POINTS | TASK | POINTS | OF POINTS |
|---|---|---|---|---|---|
| 1 | | | 3 | | |
| 2 | | | 4 | | |

IME IN PRIIMEK: _____

ŠTUDENTSKA ŠTEVILKA: _____

DATUM: _____

PODPIS: _____

**1. naloga:** Peter Puzzle has found a small sheet of paper with the following integers

$$5, 8, 66, 31, 27, 75, 29, 62, 36. \tag{1}$$

Peter has come up with a game where he randomly generates $m$ integers and wonders if these numbers are *contained* in the same order between integers (1). For example, for $m = 2$ and integers 27 and 29 the answer is YES, while for 29 and 27 the answer is NO.

QUESTIONS:

A) (i) For integers (1), what is the maximal possible $m$ such that the answer can still be YES? Justify your answer. (ii) Write down an algorithm that, given integers $N$ and generated integers $M$ determines if integers $M$ are contained in $N$. (iii) What kind of algorithm design technique have you used? (iv) What is the time complexity of your algorithm? Justify your answer.

B) Let us change the game a bit. We have $n$ integers $N$ and $m$ integers $M$ and now we are looking for the longest sequence of integers contained in both $N$ and $M$. (i) Let $N$ be integers (1) and let $M$ be integers 65, 66, 60, 27, 29, 88. Find the longest sequence of integers contained in both $N$ and $M$. (ii) Write down an algorithm that finds such a sequence for arbitrary $N$ and $M$. (iii) What kind of algorithm design technique have you used?

C) What is the time complexity of your algorithm? Justify your answer.

**2. naloga:** Trees. During the lectures, we have learned about three types of tree traversals. In this problem, we only deal with binary trees.

QUESTIONS:

A) (i) Does there exist a binary tree whose *in-order* traversal gives rise to a sequence of integers (1)? Justify your answer. (ii) Does there exist a binary tree whose *post-order* traversal gives rise to a sequence of integers (1)? Justify your answer.

B) (i) Is it possible that *pre-order* traversals of two different binary search trees give rise to the same sequence of integers? Justify your answer. (ii) Suppose that we are given pre-order and in-order traversals of a tree. Write down an algorithm that based on given traversals reconstructs the tree.

C) What is the time complexity of your algorithm? Justify your answer.

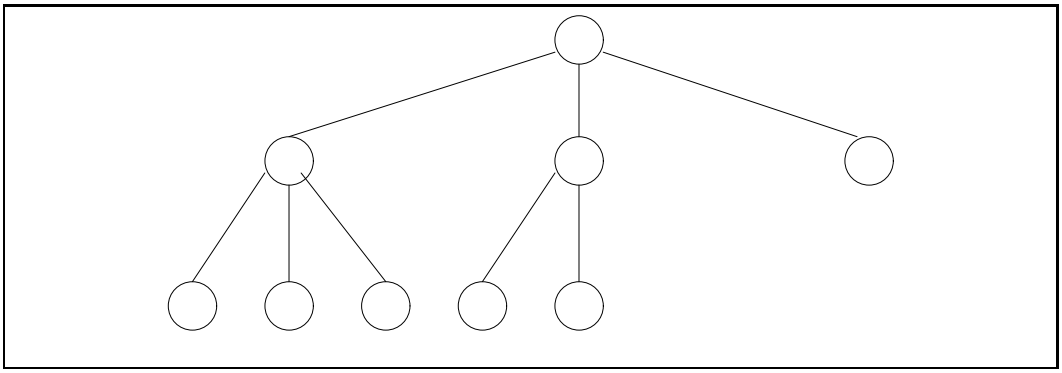

**Figure 1:** A ternary heap.

**3. naloga:** Priority queues. In general, heaps are $k$-ary, while during the lectures we have learned about binary heaps. An example of a ternary heap is given in Figure 1. Let this be a min heap, that is, in the root of any subheap we have the minimal element of this subheap.

QUESTIONS:

A) (i) Insert values (1) into the heap given in Figure 1. Draw the final heap. (ii) Perform the operation `DelMin` over the obtained heap and draw the final picture. Perform the operation by sinking down the gap and then raise up the element if needed. (iii) Perform the operation `DelMin` over the heap from (i) again and draw the final picture. This time perform the operation by moving an element from a leaf into the root of the heap and then sink it down as much as needed.

B) (i) Exactly how many comparisons have you made when performing deletion (ii) from question A and how many when deletion (iii)? (ii) Suppose that we have $n$ elements in a ternary heap. How many comparisons is needed when deleting by sinking down the gap? Justify your answer.

   HINT: Note that the gap first sinks down to the bottom and then the element flows up.

C) So far we have considered ternary heaps, but from now on let us consider $k$-ary heaps for an arbitrary $k$. (i) How many comparisons is needed when deleting by sinking down the gap in the worst case? Justify your answer in terms of the parameters $n$ and $k$. (ii) What is the optimal $k$? Justify your answer.

**4. naloga:** Graph algorithms. We define two operations on graphs. The first one is `Shorten(G)` that makes from a graph $G(V, E)$ a new graph $G'(V, E')$ with the

edge $(u, v)$ in $E'$ if and only if there exists $x \in V$ such that both edges $(u, x), (x, v)$ are in $E$. The second operation is `Add(G1, G2)` that adds two graphs $G_1(V, E_1)$ and $G_2(V, E_2)$ to obtain a new graph $G_3(V, E_3)$, where $(u, v) \in E_3$ if $(u, v) \in E_1$ or $(u, v) \in E_2$. An example of both operations is given in Figure 2. In this

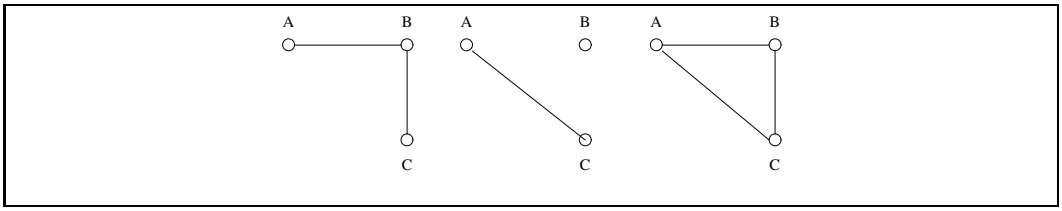


**Figure 2:** The operation `Shorten` on the left graph returns the graph in the middle, while `Add` on the left and the middle one returns the right graph.

problem, all graphs are to be represented by adjacency matrices.

QUESTIONS:

A) (i) Represent $G_1(V, E)$ in Figure 3 by an adjacency matrix. (ii) On $G_1(V, E)$

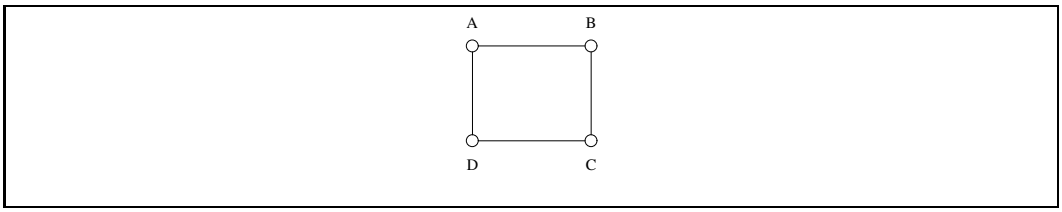


**Figure 3:** A graph $G_1(V, E)$ with $n = 4$ nodes.

preform the operation `Shorten` and draw the obtained graph. Write down its adjacency matrix.

B) (i) Write the function `Shorten` that returns a graph as described above. (ii.) What is the time complexity of your algorithm? Justify your answer. (iii.) Write the function `Add(G1, G2)` that returns a graph as described above. (iv.) What is the time complexity of your algorithm? Justify your answer.

C) Suppose that we have successfully implemented both operations and that we run the following algorithm ($n$ is the number of nodes in the graph $G$):

```
WhatMakes(G):
  for i= 1 ... n do:
    tmp= Shorter(G)
    G= Add(G, tmp)
  return G
```

What the graph returned by the function `WhatMakes` looks like? Justify your answer.