Algoritmi in podatkovne strukture – 2
First Midterm (2016/17)

This test must be taken individually. Any and all literature may be used while taking this test. Answer diligently *all* questions.

Bonus points might be awarded if you at least partially correctly answer each question.

Duration of the test: 60 minutes.

We wish you a lot of success – veliko uspeha!

NALOGA	TOČK	OD TOČK	NALOGA	TOČK	OD TOČK
1			3		
2			4		

IME IN PRIIMEK: _____

ŠTUDENTSKA ŠTEVILKA: _____

DATUM: _____

PODPIS: _____

1. naloga:

Dictionary – basics. Suppose we have the following numbers sorted in increasing order: 1, 4, 19, 19, 19, 24, 39, 40, 42, 46, 50, 53, 56, 58, 59, 61, 66, 69, 74, 77, 79 in 90.

VPRAŠANJA:

A) Draw a binary search tree that has the minimal height and contains all the above elements.

B) (i) Write down an algorithm, which from n *different* sorted numbers in an array `A` constructs a binary search tree with the minimal height. Show that your tree has the minimal height. (ii) What is the time complexity of your algorithm? Elaborate.

C) Propose a data structure (or structures), which will be the least time consuming for the following operations: (i) first we insert n elements (`Insert`); and (ii) then we do n^2 queries (`Find`). How much time does your structure (or structures) spend for all operations? Elaborate.

HINT: The better your solution, the more points you will earn.

2. naloga: *Dictionary – more advanced.* Peter Zmeda is doing his homework and he has got the following numbers in a random order: 69, 74, 58, 4, 61, 90, 19, 46, 1, 19, 40, 59, 39, 56, 19, 50, 42 in 53. Since he will be inserting numbers in a skip list, he also needs random values 0 and 1:

1 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 1 0 1 1
1 0 0 0 0 0 1 1

VPRAŠANJA:

A) How does Peter's skip list look like after inserting all numbers? Consider that the height of an element is the number of successive ones plus 1: for example, the first inserted element, which is 69, will have the height 2, since one is followed by zero (underlined first 0 and 1 above). When inserting the next number, which is 74, use the next one and zero and so on.

B) Let us define the function `Left(x)` over a dictionary, that returns the biggest element in a dictionary which is smaller then `x`. Peter should implement this new function. Help him.

(i) Describe how does this new function work[1]. (ii) What is the time complexity of your algorithm? Justify your answer. (iii) Is it possible to improve

[1] Pseudocode or algorithm will earn you more points.

(modify) a skip list in order to speed up the search of the left element? Justify your answer.

C) Peter's teacher has a lot of ideas, and then Peter has to implement them. This time he proposed to implement a dictionary with a hash function instead of using a skip list, because he read somewhere that a hash function is more efficient. (i) How should now Peter implement the function `Left(x)`? (ii) What is the time complexity of the implementation? Justify your answer. (iii) Is it now possible to do anything so that the implementation is faster? Justify your answer.

3. naloga: Suppose that we have the universal set $\{1, 2, ..., 16\}$ of numbers, and that every number represents a separate independent set.

VPRAŠANJA:

A) Simulate the following operations over the set of sets: `Union(1, 7)`,`Union(2, 9)`, `Union(3, 9)`,`Union(15, 10)`,`Union(16, 1)`,`Union(12, 12)` and `Union(13, 7)`. Write down the resulting sets.

B) And then again Peter's teacher. This time he wants Peter to implement a data structure, which should support apart from operations `Union` and `Find` discussed on the lectures in the context of disjoint sets, also the operation `List(x)` which returns all elements of the set to which `x` belong. (i) Describe this new data structure. (ii) What is the time complexity of all tree functions? Justify your answer.

C) In the disjoint set data structure, each element is a member of exactly one set. Suppose we allow the element to be a member of at most two sets. (i) Describe the corresponding data structure which supports efficient operations `Union()` in `Find()`. (ii) Describe both operations and justify their correctness, time, and space complexity. (iii) Do you notice any conceptual issue of an element being a member of more than one set? How would you deal with it?

HINT: What exactly does `Find()` return?

4. naloga: *Priority queues, select, and rank*

VPRAŠANJA:

A) Assume 18 random numbers from eq. (**??**). (i) Insert the first 10 numbers into lazy binomial heap H_1. Draw the obtained H_1 and write down the number of performed operations after each action, and the overall number of operations. (ii) Insert the remaining 8 numbers into another lazy binomial heap H_2 and write down the number of performed operations respectfully. (iii) Call function `DelMin(`H_2`)`, draw the obtained H_2 and write down the number of performed operations. (iv) Merge two binomial heaps H_1 and H_2, draw the resulting data structure, and write down the number of performed operations.

   HINT: For every moment, you need to know what is the smallest element in the heap.

B) When searching for the median element we divided the input list into groups of 5 elements. (i) Could we divide it to groups of 7 elements – would the algorithm still work? (ii) What if we divided the input list to groups of eleven – would the algorithm still work? Elaborate.

C) Rank slightly differently. Peter Puzzle already has n_1 elements stored in a sorted array. Afterwards he obtains n_2 new elements which he stores into a hash table. Finally he needs to compute the `Rank(x)` request. (i) Describe the procedure and the data structure for efficiently completing his task. (ii) Justify the correctness of your solution. (iii) Evaluate the time and space complexity of your solution.

   HINT: Your approach will most likely depend on parameters n_1 and n_2.