Data Structures and Algorithms (2020/21) Written exam February 10th, 2021

Written exam must be taken individually. Any and all literature may be used while taking the exam. In your answers be precise, and: (i) answer the questions *as they were asked*; (ii) answer *all* tasks – if you will be answering to *all* tasks you might get bonus points; and (iii) when you are asked to provide the justification of correctness of your answer provide it as otherwise the points will be deducted.

Time: 105 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. task: Router is a device with k ports[1]. When an IP package arrives at one of the ports, the router needs to decide which port should the package be routed to. Information on which package should be routed where is stored in a data structure called *the routing table*. An example of a small routing table:

Destination	Gateway	Flags	Netif	Expire
default	192.168.2.1	UGS	em0	
127.0.0.1	link#3	UH	lo0	
192.168.2.155/25	link#1	U	em0	
192.168.2.101	link#1	UHS	lo0	
192.168.2.0/23	link#2	U	igb0	
192.168.126.1	link#2	UHS	lo0	

We can implement the routing table using a trie over alphabet $\Sigma = \{0, 1\}$ where each key represents a subnet address and its corresponding value the target port (interface, `Netif` in the table above), where the subnet is accessible on. For example, in the table above we have keys [2]

```
192.168.2.155/25 = 11000000 10101000 00000010 1.......
192.168.2.0/23 = 11000000 10101000 0000001.  ........
```

with respective values `em0` and `igb0`. The package is routed to the interface which matches best its IP address. For example, if a package with destination IP 192.168.2.140 arrives, it will be routed to interface `em0`, because it matches best the first key. And if another package with destination IP 192.168.2.14 arrives, it will be routed to interface `igb0`.

QUESTIONS:

A) Build a trie using path compression (*PATRICIA*) from the following keys and values[3]

(136/1, A)	(138/8, B)	(16/7, C)	(162/8, È)	(92/5, D)
(32/4, E)	(91/2, F)	(10/6, G)	(82/3, H)	(47/3, J)

B) Consider the use of the level compression on the trie from the previous task. Argue its feasibility.

C) Suppose a package arrives with destination IP d. Write down an algorithm for routing the package to an interface using a routing table implemented with a trie.

[1] k can be as small as 4 and up to dozens

[2] Blanks are only for the sake of brevity, we store a string of 25 zeros and ones because 25 is the mask length.

[3] To make the task easier, all keys are 8 bits long and values are single letters.

2. task: We have the following numbers

$$59, 29, 93, 40, 40, 21, 82, 37, 36, 82, 15, 38, 6, 8, 5, 86, 68, \quad (1)$$

stored in array `S` counting from 0. We define function `Max(S, i, j)` which returns the maximum element in array `S` between index i and j inclusive. For example, `Max(S, 7, 12)` returns 82.

QUESTIONS:

A) (i) Implement `Max(S,i,j)` for arbitrary i and j running in $O(1)$ time. You are allowed to use *any preprocessing* beforehand. (ii) What is the space complexity of your solution? Elaborate.

B) Basic operation on an array is accessing the i-th element – for example `S[4]` corresponds to the fifth element in array `S`. An array is usually implemented as an implicit data structure. However, we could implement it differently by using a skip list – but the elements should not be sorted by their value. For example, using the elements from (1), the first element should be 59, then 29, and so on. (i) Describe the procedure to access the i-th element as described above in the skip list. (ii) What is the time complexity of your procedure?

C) We extend the set of operations with `Insert(S, i, e)` which inserts an element e to position i and shifts the remaining elements for one slot. If we call `Insert(S, 1, 77)` on our data (1), then we will have 59 at position 0, 77 at position 1, 29 at position 2, and so on up to 46 at position 25. (i) Describe an efficient implementation of the data structure which supports operations `Max()` and `Insert()`. (ii) Can operation `Max()` run in time $o(\log n)$, where n is the number of elements stored in the array? Elaborate.

3. task: Peter Zmeda is about to do his homework. He needs to solve seven tasks and for each task he earns a number of points. Also, he estimates to spend the following amount of time for each task:

points	7	9	5	12	14	6	12
time (in h)	3	4	2	6	7	3	5

To complete his homework, Peter has 15 hours of left. Obviously he cannot complete all the tasks, so he needs to decide which tasks to take in order to maximize the number of points. This is an optimization problem.

QUESTIONS:

A) Suppose Peter earns partial points for partially completed tasks. This means, if he solves the half of the first task, he spends 1.5h and earns 3.5 points. (i) Which tasks should Peter solve and to what extent in order to earn as much points as possible? (ii) Justify the correctness of your approach.

B) Suppose we can solve tasks partially. Let us have n tasks, each earning p_i points, requiring t_i time, and the time limit T. (i.) Write down an algorithm which yields an optimal solution. (ii.) Justify the correctness of your algorithm. (iii.) What are the time and space complexity of your algorithm? Elaborate.

C) Now suppose Peter will earn points only, if he completes a task (the principle of all or nothing). (i) Which tasks from the table above should Peter take now? (ii) Justify the correctness of your answer. (iii.) In general, what if there are n tasks to consider and we need to find an optimal solution acknowledging the all or nothing principle? Write down the algorithm.

4. task: Na Figure 1 there is a directed graph on the left containing a cycle $BCED$

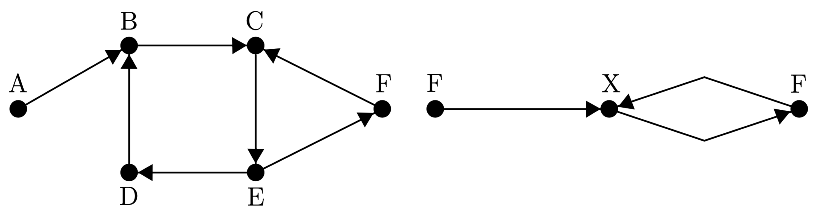

Figure 1: Sesedanje ciklov.

which then collapses into a single node X as seen on the right. In this task we have **directed** graph $G(V, E)$, where $|V| = n$ and $|E| = m$. THe graph G is defined by an adjacency matrix, while vertices are labeled by numbers between 1 and n.

QUESTIONS:

A) Let $A \in V$. (i.) Write down algorithm `Cikel(`G`,` A`)` which finds a cycle in graph G containing node A and returns a list of nodes c in that cycle. (ii.) What is the time complexity of your algorithm? Elaborate. (iii.) Does an algorithm which runs in time $o(m)$ exist? Elaborate.

B) In graph G there is at most one directed edge from A to B (not a multigraph). (i.) Write down algorithm `Sesedi(`G`,` c`)`, which collapses a cycle c in graph G into a single node, where c is the return value of `Cikel(`G`,` A`)`.

(ii.) What is the time complexity of your algorithm? Elaborate. (iii.) Functions `Cikel()` and `Sesedi()` have already been implemented. Compose an algorithm which collapses all nodes in graph G.

C) We are aware that finding a Hamiltonian cycle in arbitrary graph is an NP-complete problem. Peter Zmeda solved the previous task and got na idea how to solve the Hamiltonian cycle problem: he needs to collapse all cycles in the graph, and if, eventually, only a single point remains, he has found a Hamiltonian cycle. (i.) Draw an example of a graph where his approach works. (ii.) In general, why is his approach flawed? Elaborate by finding a counter exemplar graph.