Data Structures and Algorithms
Spring 2021
Assignment №2

IT University Innopolis

Contents

1 About this homework

This homework assignment only has theoretical exercises.

Solutions to theoretical exercises have to be elaborate and clear. Write all solutions in a single document, then save or compile it as a PDF for submitting.

Do not forget to include your name in the document!

1.1 Submission

You are required to submit your solutions to all parts via Moodle.

For this assignment you will need to submit:

- a PDF file with solutions for theoretical parts.

Submit a single file with your name and group number. For example, if your name is *Ivanov Ivan* and your group is *BS20-05* then your file should be named `BS20-05_Ivanov_Ivan.pdf`.

2 Theoretical part

2.1 Dynamic programming

Two cities A and B are connected by a highway. Your task is to place shopping centers along the highway in an efficient way, maximizing estimated revenue. Possible locations of shopping centers along the highway are given by N numbers $x_1, x_2, \ldots, x_N$. Each number specifies the number of kilometers from city A to that location on a highway. For each position x_i we also know in advance the estimated revenue r_i that we get if we place a shopping center there. There is a restriction that no two shopping centers can be placed within d kilometers of each other. The highway length is H kilometers and we have $0 < x_1 < x_2 < \ldots < x_n < H$.

For example, if we have 3 locations $x_1 = 5, x_2 = 10, x_3 = 14, x_4 = 15$ with estimated revenue $r_1 = 80, r_2 = 150, r_3 = 60, r_4 = 75$ and $d = 6$ then the maximum revenue is 155 when we place two shopping center at locations $x_1 = 80, x_4 = 15$.

Describe a general algorithm for any number N of available locations, any length of the highway H:

1. Write down pseudocode for a recursive algorithm that solves the problem.
2. Provide asymptotic worst-case time complexity of the recursive algorithm.
3. Identify overlapping subproblems.
4. Write down pseudocode for the optimized algorithm that solves the problem using dynamic programming (top-down or bottom-up). The algorithm should compute both the maximum estimated revenue **and** the specific locations where shopping centers should be placed.
5. Provide asymptotic worst-case time complexity of the dynamic programming algorithm.

2.2 Sorting

1. Briefly explain how insertion sort works.
2. Prove that the worst and the best case running times of insertion sort are $\Theta(n^2)$ and $\Theta(n)$, respectively.
3. What is a k-sorted array? Is insertion sort fast or slow, relative to its worst-case, when applied to a k-sorted array? Justify your answer by computing the running time of insertion sort for a k-sorted array.
4. Implement a comparison-based sorting algorithm to sort the following table of data, containing information on events (`code`, `start_date`, `end_date`, `sponsor`, `description`). Given the `code` as the key, provide a pseudocode for comparing two keys that will be used in your sorting algorithm.

code	date_start	date_end	sponsor	description
A001	20-03-2021	27-03-2021	IU	...
A123	20-04-2021	20-05-2021	DS	...
B001	10-04-2021	17-04-2021	MTS	...
A009	12-04-2021	15-04-2021	GTK	...

5. Write two versions, one recursive, the other iterative, of a boolean function named `belongs` which takes as arguments an array `A` of ordered integers, a start index `from`, an end index `to` and an integer `x` to search for and which returns `true` if `x` is in the array `A` between index `from` and index `to`, and `false` otherwise.

6. Write a variant of the previous function called `search` which takes the same arguments as `belongs`, and which returns the index `i` of the cell containing `x` if `x` appears in the array `A` between the index `from` and the index `to` and `null` otherwise.

7. Establish the recurrence relation for the running time of `search` and determine the time complexity by applying Master Theorem.

8. Explain the benefit of using `search` in the context of insertion sorting. Give a version of insertion sorting using `search`. How does it affect the time complexity of insertion sort?

9. Is Bubble Sort difficult to parallelize? Why? Provide pseudocode (and explain) a parallel version of bubble sort.

2.3 Balanced Binary Search Trees

Consider the following sequence of numbers:

$$25, 60, 35, 10, 5, 20, 65, 45, 70, 40, 50, 55, 30, 15$$

1. Explain in words the properties of AVL trees.
2. Explain in words or in pseudocode `insert` operation for AVL trees.
3. Considering the following tree as the initial state of the tree, construct the AVL tree by adding values in the order of the sequence above. Particular attention should be paid to explaining the reasoning.

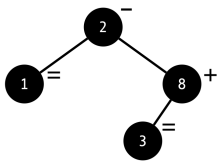

Figure 1: Initial AVL tree for insertion.

4. List values at the nodes of the tree by in-order traversal of the tree. What property does resulting sequence have? Is this always the case?
5. Starting with the same initial tree, construct a 2-3 tree by inserting values in the order of the sequence above.
6. Explain in words or in pseudocode `delete` operation for AVL trees.
7. Give the trees obtained by deleting 45 and then 30 in the tree below.

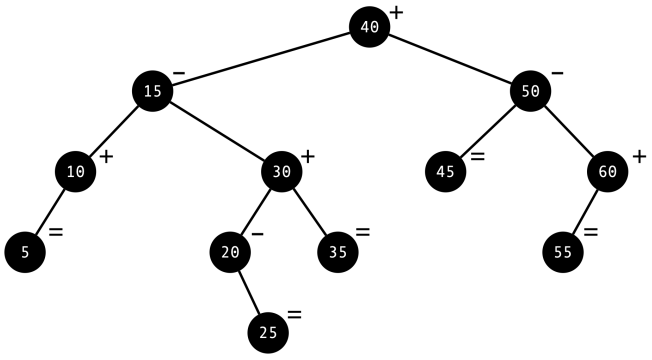

Figure 2: Initial AVL tree for deletion.

2.4 Binary Heap

Using two binary heaps describe a way to efficiently keep track of median[1] values. That is, describe a data structure that is based on two heaps and provides the following methods:

1. `insert(k)` — insert value;
2. `remove_median()` — remove median value and return it;
3. `size()` — return the size of the data structure (number of stored values);
4. `isEmpty()` — check if the structure is empty (no values).

Write down pseudo code for these methods, using the interface of the binary heap and provide worst-case time complexity for each method.

[1] median values are those in the middle of the sorted sequence of values. For example, 5 is median in 1, 2, 5, 100, 254. When number of elements is even, we have two median values and the algorithm can return either one.