# Sabancı University
# Faculty of Engineering and Natural Sciences

## CS301 – Algorithms

## Homework 3

Due: March 19, 2024 @ 23.55
( upload to SUCourse )

**PLEASE NOTE**:

- Provide only the requested information and nothing more. Unreadable, unintelligible, and irrelevant answers will not be considered.
- Submit only a PDF file. (-20 pts penalty for any other format)
- Not every question of this homework will be graded. We will announce the question(s) that will be graded after the submission.
- You can collaborate with your `TA/INSTRUCTOR ONLY` and discuss the solutions of the problems. However, you have to write down the solutions on your own.
- Plagiarism will not be tolerated.

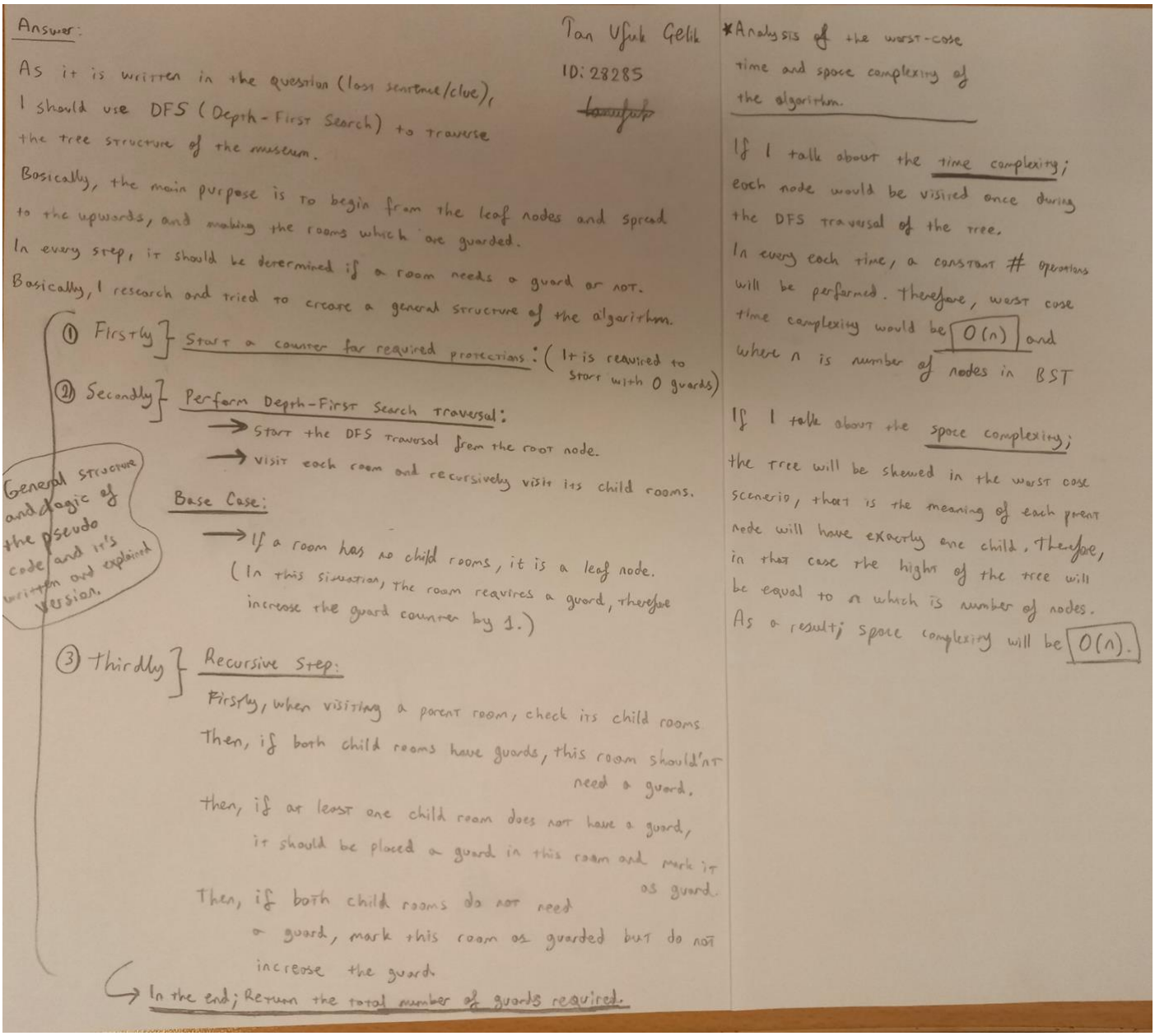
**Late Submission Policy**:

- Your homework grade will be decided by multiplying what you normally get from your answers by a "submission time factor (STF)".
- If you submit on time (i.e. before the deadline), your STF is 1. So, you don't lose anything.
- If you submit late, you will lose 0.01 of your STF for every 5 mins of delay.
- We will not accept any homework later than 500 mins after the deadline.
- SUCourse's timestamp will be used for STF computation.
- If you submit multiple times, the last submission time will be used.

**Question 1**

Consider a uniquely designed museum where rooms are arranged in a tree structure. Each room can have up to two child rooms connected by a path. Your task is to develop an algorithm to place a minimum number of security guards so that the entire museum is guarded. A guard placed in a room can guard that room, its parent room, and its direct child rooms.

(a) Develop an algorithm to find the minimum number of security guards required for any given museum structured as a standard binary tree. Analyze the worst-case time and space complexity of your algorithm.
**Hint:** Consider using DFS (for a bottom–up traversal of the rooms).



Answer:

Tan Ufuk Gelik
ID: 28285

As it is written in the question (last sentence/clue), I should use DFS (Depth-First Search) to traverse the tree structure of the museum.

Basically, the main purpose is to begin from the leaf nodes and spread to the upwards, and making the rooms which are guarded.

In every step, it should be determined if a room needs a guard or not.

Basically, I research and tried to create a general structure of the algorithm.

① Firstly } Start a counter for required protections: ( It is required to start with 0 guards)

② Secondly } Perform Depth-First Search Traversal:
→ Start the DFS traversal from the root node.
→ Visit each room and recursively visit its child rooms.

General structure and logic of the pseudo code and it's written and explained version.

Base Case:
→ If a room has no child rooms, it is a leaf node.
(In this situation, the room requires a guard, therefore increase the guard counter by 1.)

③ Thirdly } Recursive Step:
Firstly, when visiting a parent room, check its child rooms
Then, if both child rooms have guards, this room should'nt need a guard.
then, if at least one child room does not have a guard, it should be placed a guard in this room and mark it as guard.
Then, if both child rooms do not need a guard, mark this room as guarded but do not increase the guard.

↳ In the end; Return the total number of guards required.

*Analysis of the worst-case time and space complexity of the algorithm.

If I talk about the time complexity; each node would be visited once during the DFS traversal of the tree.
In every each time, a constant # operations will be performed. Therefore, worst case time complexity would be O(n) and where n is number of nodes in BST

If I talk about the space complexity; the tree will be skewed in the worst case scenerio, that is the meaning of each parent node will have exactly one child, Therefore, in that case the hight of the tree will be equal to n which is number of nodes.
As a result; space complexity will be O(n).

(b) Discuss the alterations needed in the algorithm, as well as the changes in worst-case time and space complexity when the museum structure is known to be a red-black tree.

Answer

Tan Yusuf Gelik
ID: 28285

- The tree would be self-balancing, when the museum structure is known to be RBT. This balancing could affect the worst height of the tree. This situation makes it more consistent than an unbalanced binary tree. Because it operates on the tree's structure and NOT balancing and color properties, the algorithm does not need to be specifically altered for the Red-Black Tree properties.

If I talk about the worst-case time complexity;

Each node is visited once, nevertheless of the tree's balancing properties, it will be $O(n)$, and also where n is the number of nodes in the tree.

If I talk about the worst-case space complexity;

the space complexity would be $O(\log n)$, where n is the number of nodes. Since, worst case space complexity is is determined by the tree's height, can see a more consistent performance in a Red-Black tree.

(c) Given that each room has a number, from the viewpoint of a visitor intending to find/visit room X in the museum starting from the entrance room (i.e., root node), explain the differences experienced when the museum structure is a standard binary search tree versus a red-black tree.

Answer;

Tan Ufuk Gelik
ID: 28285

RBT ensures a more efficient search since they are balanced by definition, in the searching a specific room(node).
More detailly; in the worst case, the standart binary tree is skewed and it is highly inbalanced, which could be degenerated into a linked list and leading to degraded search performance with a time complexity of $O(n)$.
But also, the Red-Black tree guarantees logarithmic height even in worst cases which has a time complexity of $O(\log n)$.

While searching in a BST can lead to highly variable path lengths and search times, searching in a Red-Black Trees offers a more efficient and consistent path due to the tree's balanced nature.
Because it would have a more efficient and predictible structure, a museum structured in RBT would enhance the visitors' experience.

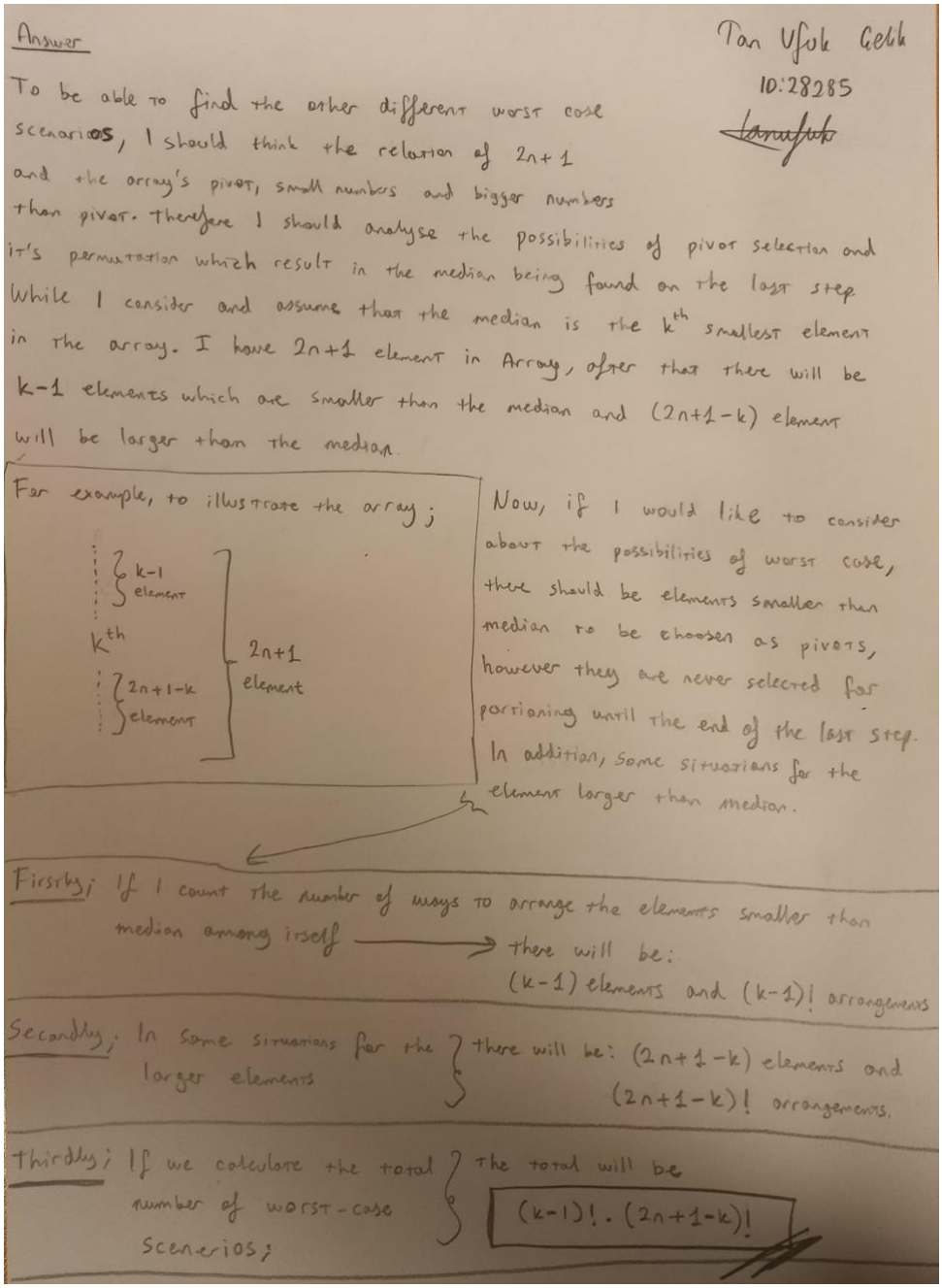
**Question 2**

We are given an array $A$ with $2n + 1$ distinct elements. Suppose that we are using the randomized selection algorithm to find the median. In a worst–case scenario of this algorithm, the median is found at the very last step, where each step before that gets rid of only one element in the array. How many different worst–case scenarios are there for finding the median in $A$?

Example: Suppose that we have $A = [4, 5, 1, 3, 2]$.

One of the worst–case scenarios is to pick the following elements as the pivots in this order $1, 2, 5, 4$. Because (randomly but unluckily) if we pick these numbers in this order as the pivots, we will get rid of only the pivot in each step, and only at the very end (when we have only element 3 remaining, which is the median of the original input array) we will find the median.

The other worst–case scenarios are:
1,5,2,4
1,5,4,2
5,1,4,2
5,1,2,4
5,4,1,2



**Question 3**

In the WCL Select algorithm, suppose that we modify the approach to partition the array into groups of size 2k+1 instead of the usual groups of 5. Write down the recurrence for the running time of the algorithm for this general case.

Answer:

Tan Ufuk Gelik
ID: 28285

The WCL → Worst Case linear

the wcl algorithm, generally associated with the median-of-medians approach, and it is a selection algorithm for finding the smallest/largest element in an unordered list. Also, it is designed to have worst-case linear time complexity. therefore, this makes it especially useful for scenarios where predictable performance is required. Algorithm typically divides the array into groups of fixed size, finds the median of each group, and then recursively applies the same process to the medians to find a "good" pivot. This pivot is used to split the array into smaller pieces and the algorithm iterates on one of them.

In normally, the original WCL select algorithm divides the array into groups 5 element, and finds the median of each group, and then uses these medians to choose a pivot. The pivot partitions the array into smaller arrays, where the selection algorithm is recursively applied.

However, in this question, we should modify the group size to $2k+1$, where k is a positive integer, the structure of the algorithm should be change and also effects the recurrence relation for it's running time.

→ The new reccurrence relation for the Algorithm's running time, and with $2k+1$ group size can shown as;

Recurrence for the running time:

$$T(n) = T\left(\frac{n}{2k+1}\right) + T(\alpha n) + O(n)$$

we can basically and generally say that it is $T(n)$

for example;
$\frac{n}{5}$ → constant } so, it is $O(1)$
But;
$\frac{n}{2k+1}$ } $O\left(\frac{n}{2k+1}\right)$

$T\left(\frac{n}{2k+1}\right)$ } It represents the time to find the median of the $2k+1$ sized group's medians.

$T(\alpha n)$ } $\alpha$ → constant less than 1.
It is for the recursive call on the reduced problem size after partitioning with the chosen pivot.
} The exact value's of "$\alpha$" depends on the efficiency of the pivot selection, which is influed by the choice of k.

$O(n)$ } It contains the linear work done outside the recursive calls.