CME 2204 ALGORITHM ANALYSIS

Greedy Approach Assignment

Report

Completed Tasks:

I have completed all the tasks requested from us in this project. The tasks I have completed are as follows: I read the yearly player demands and player salary files given to us and assigned them to the arrays in my program. Using Greedy Approach, I wrote a function that selects the best possibility in its current situation and obtains results close to the best result with this method. By using this function, I printed the result of the greedy approach that we obtained in accordance with the information and parameters given to us.

Uncompleted Tasks:

There is no uncompleted task in the code, all parts are completed on time. In addition, the code has been tested with different parameters to eliminate errors.

Run-Time Complexity:

When we examine the code to calculate the run-time complexity, we see that the main structure of the code consists of a loop that examines all the years one by one. For each year in this loop, the next year is examined, and the current best case is selected and continued. A new loop is created to find the best situation by examining the next year. This loop is repeated as much as the difference between the number of players we can produce for free (p) and the player demands of the next year. When we examine the code in this way, the outer loop repeats n times, so that n = number of years. The inner loop, on the other hand, repeats up to p at most, since it will repeat up to the number of players required for the next year. In this case, since operations such as; assignment, addition, if else blocks work as O(1), the runtime complexity of the code is calculated as **O(n * p)**.

Space Complexity:

If we examine the space complexity of the code, I directly print the best case found at each step to the screen, and increase the cost by adding to it at each step. Therefore, extra storage is not used to find the minimum cost or to show the best situation for each year. As a result, the space complexity of the code becomes **O(1)** because it does not need an extra memory to work.

Comparison of Dynamic Programming and Greedy Approach:

If I am going to compare dynamic programming and greedy approach, I need to compare the runtime complexities, space complexities of these algorithms and the accuracy of their results.

Firstly, if I compare the run-time complexities of both algorithms. In Dynamic Programming, every probability that may occur for each year is compared respectively and the best probabilities are kept in an array. In the Greedy Approach, we look at the possibilities that may occur after just one year and proceed by selecting the best possibility. Since Dynamic Programming needs to choose the best probability for all years, it needs to try more possibilities. For this reason, the run-time complexity of dynamic programming is higher than greedy approach. So, greedy approach run faster than Dynamic Programming.

Secondly, if we compare the space complexity of both algorithms. In Dynamic Programming, the best probabilities for all years are kept in a array. In addition, a trace back array is used to keep track of which probability we choose each year. In Greedy Approach, we don't need an array to keep probabilities, as we only select the best possibility based on the next year. Greedy Approach works without using extra memory. As a result, Greedy Approach's space complexity is less than the space complexity of Dynamic Programming, so the Greedy Approach is more memory efficient than Dynamic Programming.

Finally, if we compare the results of both algorithms. Dynamic Programming gives the best result because it selects the best cases considering all the years. In Greedy Approach, on the other hand, since only looking at the next year, that is, the best case for the current situation is selected, it gives results close to the best result, but we cannot say that it always gives the best result.

Consequently, when we compare the two algorithms in general, we observe that Greedy Approach works faster and without using extra memory. In other words, we can say that Greedy Approach works more efficiently than Dynamic Programming. But on the other hand, when we look at the results of the algorithms, we observe that Dynamic Programming clearly achieves the best result, while Greedy Approach achieves results close to the best result. As a result, if we care about efficiency, we should use the Greedy Approach or use Dynamic Programming if we want to achieve clear and precise results.