DOKUZ EYLÜL UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

CME 2204
ALGORITHM ANALYSIS
ASSIGNMENT – 2

Dynamic Programming & Greedy Approach

ELİF ÖZKER
2019510094

Introduction

The hero class was created because every hero has common features and then added attributes such as name and status. Then, the file operation part was started. After the file was read in this part, the hero's properties were split and assigned to its own list. Then, according to the max level selection entered by the user according to selection, a new single list was created and the heroes at the allowed levels were thrown into it and then worked on this list in the approaches

Dynammic Programming Approach

I tried to adapt the knapsack algorithm in the dynamic programming part. And then I would check the selected heroes according to the unselected hero status and print them but this code is partially working. In some cases, he can make the best choice, and sometimes he can't. I think it's because of the control I put in not to add the same hero status.

Greedy Approach

In the Greedy approach, I used the attack point / gold ratio. I said cost as the result of this operation and added it as the attribute of the hero class. Later, I sorted the objects in the method according to their cost only and allowed the algorithm to choose accordingly. I used a comparator for sort. Since the comparator itself is sorted from smallest to largest, I had to use reversed and it is sorted from largest to smallest.

Random Approach

In the random approach, how many soldiers will be selected from which level (of course, according to the allowed level size) are assigned randomly. If we do not have enough money, a random number is assigned again. Finally, if we do not have enough money for all heroes, they exit the process using flags.

RUNNING TIME ANALYSIS

Dynamic programming is fast as we expected. This is because it doesn't look at subproblems over and over, because it can calculate them beforehand and use them later.

FOR GOLD_AMOUNT = 440
FOR MAX_LEVEL_ALLOWED = 9
FOR NUMBER_OF_AVAILABLE_PIECES_PER_LEVEL=7

On average, we see these values.

- **Dynammic Programming Approach:** 4.8598515
- **Greedy Approach:** 5.7559295
- **Random Approach:** 8,84735

REFERENCES

[1] https://www.geeksforgeeks.org/printing-items-01-knapsack/