Assignment-2

Random Approach Part:

In the Random part, I opened a random function as much as the number of data we have and determined the pieces in this way. I checked with a boolean array if there were any pieces it could take and made it continue or stop working accordingly. And of course, I did the necessary checks (level, number of pieces, etc.) in the meantime. That was actually the random part.

Greedy Approach Part:

In the greedy approach part, I created an algorithm based on the attack point / gold ratio that made the most sense to me. First, it calculates the odds for all the pieces to be used. Then it keeps the idex of the one with the largest ratio in an array. and this index throws the stone in the game temporarily out of the game. After deactivating other necessary stones of the same type, it repeats the same process again. It continues in this way until there is no piece left to buy with the gold in hand.

Dynamic Programming Part:

In the dynamic programming part, I could not solve some problems. I realized that the most suitable algorithm for this problem is the knapsack algorithm, but I could not use the algorithm fully. The first problem is selecting certain tiles, but I couldn't quite keep track of which gems he chose. Another issue is a level related issue. I could not fully adapt the part that is required to choose only 1 stone from the same level. Other than that, there is no problem, I tried to write as general code as possible, I only had a problem in this adaptation part. The algorithm I wrote before, on the other hand, works in a recursive way with a logic similar to knapsack, but I chose to use it because it could not find the best result.

References:

https://www.javatpoint.com/how-to-read-csv-file-in-java

https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

Yavuz Yılmaz 2019510086