ALGORITHM HOMEWORK REPORT
KNAPSACK PROBLEM

-Problem 1 : 0-1 Knapsack problem

1)Algorithm

I implemented a greedy algorithm for this single knapsack problem.

I used arrays to store weights and values of items.Then I created new two dimensional rate array; first dimension is to indicate which item it is,that is 0 1 2 3….., second is for rates of value over weight of all items.After all rate values are found, I sort them according to these rates. After that I sent these to knapsack function as parameters and untill weight is full I fill up knapsack with items with high rate first and sum up their values recursively, additionally I created a variable c to keep how many items it takes for knapsack.Finally after finding all items and return the total value of knapsack, I write this value to output file and using variable c and first dimension of rate array I found items used and write 1s and 0s according to this to output file.

2)Closeness to optimal:

I found an algorithm for this single knapsack problem on the internet,that imlements kind of dynamic programming algorithm but not recursively indeed iteratively with aray (to run also for large input sizes)[1]. I donot know whether it is optimal or not but it was better than my solution, therefore I made comparison based on them.

For fist test input test1a	My algorithm	Other algorithm
Execution Time (just for knapsack method ,that is, finding total value) in Millis	0	8
Total Value	2649	2697
Error	$\frac{\lvert 2697-2649\rvert}{2697} * 100 = 1.78\%$	

For fist test input test1b	My algorithm	Other algorithm
Execution Time (just for knapsack method ,that is, finding total value) in Millis	0	98
Total Value	17834	18051
Error	$\frac{\lvert 18051-17834\rvert}{18051} * 100 = 1.20\%$	

For fist test input test1c	My algorithm	Other algorithm
Execution Time (just for knapsack method ,that is, finding total value) in Millis	0	1899
Total Value	146888	146919
Error	$\frac{\lvert 146919-146888\rvert}{146919} * 100 = 0.021\%$	

[1] https://www.geeksforgeeks.org/space-optimized-dp-solution-0-1-knapsack-problem/

For fist test input test1d	My algorithm	Other algorithm
Execution Time (just for knapsack method ,that is, finding total value) in Millis	0	14666
Total Value	4225353	4243395
Error	$\frac{\lvert 4243395 - 4225353 \rvert}{4243395} * 100 = 0.43\%$	

This algorithm finds a better soolution than mine (with little error rate) but the execution time is bigger. As the input size increases, this algorithm is getting slower and slower.

3)Problems that I faced during implementation:

Firstly, I tried to implement a dynamic programming algorithm. It had found the optimal solution and it was executed properly for small input sizes but for increasing number of items it couldnot find the solution because of stackoverflow. Therefore, I tried to find another way to run properly also for large sizes and I decided to implement this greedy algorithm.It runs smoothly even for large input sizes within specific, and not too long, time. But as I used arrays, it increases space efficiency a little.

I tried to use Local Search Heuristics, after taking items I tried to change some of them with others in no rules but it didnot change the solution (instead higher value it was getting smaller). Then I tried to find some rule for changing to increase the total value but I couldnot find, therefore it ended up returning just to first algorithm and not to use this one additionally.

-Problem 2: 0-1 Multiple Knapsack problem

1)Algorithm:

For multiple knapsack problem, again I tried to use same logic as in first single knapsack problem with little improvement.

For this problem again I used arrays. I found rates of all items in two dimensional rate array but for this problem I also used another two dimensional array for weights of knapsacks. After that, firstly I call knapsack function straight forward,that is no modification in weights,in this function,in addition to first problem while I am choosing items for knapsacks, instead of taking rtaes in order (0 1 2 3…) I took them as increased by number of knapsacks,that is if we have 4 knapsacks then for first knapsack items taken are: 0 4 8 ….and for second it is: 1 5 9… and so on,then, I proceed the same operations for sorted weights of knapsack,that is I sort weights as same as rates and for knapsack with largest weight I take item with largest rate and so on. Finally I compared these two total values, took larger one and write it to output file.

Besides, for output file of this problem I couldnot wr,te in desired format,that is, I find the items used for each knapsack but I couldnot write 1s and 0s accordingly. So I created 2 output files instead 1. First oneis when trying to create right format, it ended up writing only maximum of total value, and second one is to show which items are used for which knapsacks for both unsorted and sorted weights and total values of each one.

2)Closeness to optimal:

I couldnot find another algorithm for this problem, so as I donot know the optimal solutions, or at least better solution, I couldnot make comparison.

3) Problems that I faced during implementation:

It again runs in acceptable time period but as I used much more arrays I kind of increase the space efficiency of this algorithm. I tried to find another solution which is more effective but I ended up implementing it like this. Also I found and write to the another output file that which items used for which knapsack but I couldnot assign and wite 1s and 0s according to these to first output file thus I just print the total value of knapsacks for output file that should be in right format.

Kevser İLDEŞ
150116048