CS20B1044 · DAA Practice EndSem · Avinash R Chaugrani

① Logic: The algorithm we used solves the problem using greedy strategy. We first sort the profit array (As all weight are equal, we choose the maximum profit object to maximize our profit). After sorting, we choose the highest profit objects until we reach the max. threshold of knapsack.

Analysis:

Run Time for our Algo: $\{O(n\log n) + O(n)\} \rightarrow O(n\log n)$

($O(n\log n)$: (Merge) Sorting; $O(n)$: Traversing profit array)

n is Number of Objects
→ Max. Weight / (Traversing part) Weight of each object

Total Time Complexity: $O(n\log n) = O(n^2)$

Space Complexity: $O(n)$ (Arrays for Merge Sort)

---

② Logic: Here we again use greedy strategy to solve the problem (as 1). We first sort the weight array (As profits are same, we need to include objects with less weight (More no. of objects)). After sorting, we greedily choose low weight objects until we reach max. capacity of knapsack.

Analysis:

Run time for our Algo: $O(n\log n) + O(n) \rightarrow O(n\log n)$

($O(n\log n)$: (Merge sort) Sorting; $O(n)$: Traversing weight array)

Total Time Complexity = $O(n\log n) = O(n^2)$

Space Complexity: $O(n)$ (Arrays for Merge Sort)

CS20B1044 PAA Practice EndSem Avinash R Changrani

③ Logic: Our Algo solves the problem using top down approach of Dynamic Programming using the concept (technique) of Memorization where we store the recursive sub problems in a 2D array and use another function to use those subproblems and evaluate our problem and return the solution. We use memory to affect the Time complexity and get lesser time complexity.

Analysis: We run the Algo recursively and store it in a N×W table. So, the time complexity is $O(N \times W)$ (N → Number of items, W → maxweight (For smaller weights, its wrt W. polynomial))

Time Complexity: $O(N \times W)$

Space Complexity: As we maintain a N×W table. the space complexity is $O(N \times W)$