

**GREEDY ALGORITHMS**

|  |  |
| --- | --- |
| **DEPARTMENT** | **SOFTWARE ENGINEERING LEVEL ONE** |
| **COURSE** | **FUNDAMENTALS OF ALGORITHM** |
| **PRESENTED BY GROUP** | **10** |
| **COURSE SUPERVISORE** | **MR. TEKOH PALMA** |

Greedy algorithms are algorithms that find solution to a problem in the shortest time possible. That is at each step, a greedy algorithm makes a choice that seems best at the moment without considering the consequences of that choice on the future steps. Greedy algorithms are usually easy to implement and are efficient in terms of runtime. However, they may not always provide the optimal solution for every problem. That is a greedy algorithm may find a locally optimal solution, but not necessarily the globally optimal solution. Therefore, careful analysis is required to determine if a greedy approach is suitable for a specific problem. A computer scientist and mathematician called EDSGER DIJKSTRA discovered greedy algorithm. Some characteristics of a greedy algorithm include;

* The algorithm solves its problem by finding an optimal solution. This solution can be a maximum or minimum value. It makes choices based on the best option available.
* The algorithm is fast and efficient with time and complexity. Therefore, applied in solving large scale problems.
* The search for optimal solution is done without repetition. The algorithm runs once.
* It is straight forward and easy to implement.

**APPLICATION OF GREEDY ALGORITHMS**

There are various applications of greedy algorithms, some of which include;

Minimum spanning tree; this is without any cycle and with the minimum possible total edge weight. This tree is derived from a connected undirected graph with weights

Dijkstra’s shortest path; this is a search algorithm that finds the shortest path between a vertex and other vertices in a weighted graph.

Travelling salesman problem; this involves finding the shortest route that visits different places only once and returns to the starting point.

Huffman coding assigns shorter code to frequently occurring symbols and longer code to less occurring symbols. It is used to encode data efficiently.

**ADVANTAGES OF USING A GREEDY ALGORITHM**

1. Greedy algorithms are quite straight forward and are easy to implement and easy to understand. They are also very efficient and have a lower complexity time of O(N\*logN).
2. They are useful in solving optimization problems, returning or minimum value.

**DISADVANTAGES OF USING A GREEDY ALGORITHM**

1. Even though greedy algorithms are straight forward and helpful in optimization problems, they don’t offer the best solution at all times.
2. Also, greedy algorithm only runs once, so they don’t check the correctness of the result produced.

**HOW TO SOLVE FRACTION KNAPSACK PROBLEM USING GREEDY ALGORITHMS**

A knapsack has a maximum weight, and it can only accommodate a certain set of items. These have a weight and value.

The aim is to full the knapsack with the items that have the highest total values and do not exceed the maximum weight capacity.

**APPROACH TO THE PROBLEM**

There are two elements to consider: the knapsack and the items. The knapsack has a maximum weight and carries some items with high value.

**Scenario**: in a jewelry store, there are items made up of gold, silver and wood. If a jewelry thief comes to the store, they will take gold because they will make the most profit.

The thief has a bag (knapsack) that they can put these item in. but there is a limit to what the thief can carry because these items can get heavy .the idea is to pick the item that will make the highest profit sand fits in the bag (knapsack) without exceeding the maximum weight.

* The first step is to find the weight ratio of all items to know what fraction each one occupies
* We then sort this ratio in descending order (from highest to lowest). This way we can pick the ratios with the highest number first knowing that we will make profit
* After picking the highest ratio, we find the corresponding weight and add to the knapsack while keeping the following conditions in check:

**Condition1**: if the item has a lesser weight than the maximum weight of the knapsack, more items are added to the knapsack until the sum of all items in the bag is equal to the maximum weight of the knapsack.

**Condition 2**: if the sum of the items weight in the bag is more than the capacity of the knapsack, we add the fraction of the last item added. To find the fraction we do the following:

We find the sum of the remaining weights of the items in the knapsack. They must be less than the maximum capacity.

We find the difference between the maximum capacity of the knapsack and the sum of the remaining weights of the items and divide by the weight of the last item to be added.

Fraction = (maximum capacity of the knapsack – sum of remaining weights) / weight of last item to be added

To add the weight of the last item to the knapsack, we multiply the fraction by the weight.

Weight added = weight of last item to be added \* fraction

When we sum up the weights of all the items it will be equal to the knapsack’s maximum weight.

**Practical Example:**

Let’s say that the maximum capacity of the knapsack is 17, and there are three items available. The first item is gold, the second item is silver, and third item is wood.

Weight of gold is 10, the weight of silver is 6, and the weight of wood is 2

The value (profit) of gold is 40, the value (profit) of silver is 30, and the value (profit) of wood is 6.

Ratio of gold= value/weight = 40/10 = 4

Ratio of silver = value/weight=30/6 = 5

Ratio of wood = value/weight = 6/2 = 3

Arranging the ratios in descending: 5, 4, 3.

The largest ratio is 5 and we match it to the corresponding weight “6”. It points to silver.

We put silver in the knapsack first and compare it to the maximum weight which is 17. 6 is less than 17 so we have to add another item. Going back to the ratios, the second largest is “4” and it corresponds to the weight of “10” which points to gold.

Now, we put gold in the knapsack, add the weight of the silver and gold, and compare it with the knapsack weight. (6 + 10 = 16). Checking it against the maximum weight, we see that it is less. So we can take another item. We go back to the list of ratios and take the 3rd largest which is “3” and it corresponds to “2” which points to wood.

When we add wood in the knapsack, the total weight is (6 +10+2 = 18) but that is greater than our maximum weight which is 17. We take out the wood from the knapsack and we are left with gold and silver. The total sum of the two is 16 and the maximum capacity is 17. So we need a weight of 1 to make it equal. Now we apply condition 2 discussed above to find the fraction of wood to fit in the knapsack.

Frac1

Explanation of filling the remaining space in the backpack with a fractional piece of wood

Now the knapsack is filled.