Greedy Technique

Introduction

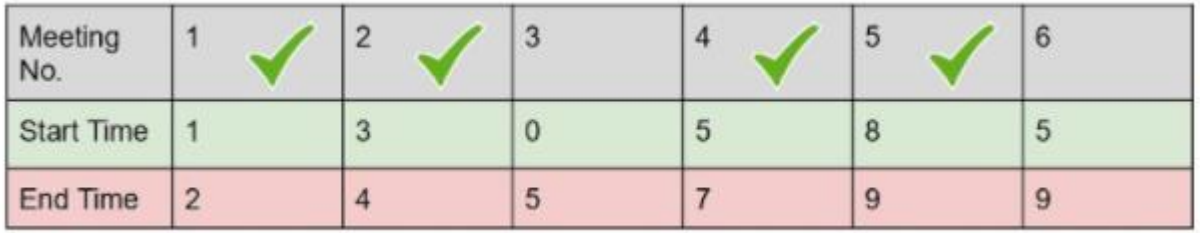
- A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.
- The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.
- This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice to produce the global best result.

- However, we can determine if the algorithm can be used with any problem if the problem has the following properties:
- **1. Greedy Choice Property**
- If an optimal solution to the problem can be found by choosing the best choice at each step without reconsidering the previous steps once chosen, the problem can be solved using a greedy approach. This property is called greedy choice property.
- **2. Optimal Substructure**
- If the optimal overall solution to the problem corresponds to the optimal solution to its subproblems, then the problem can be solved using a greedy approach. This property is called optimal substructure.

N meetings in one room

Problem Statement: There is **one** meeting room in a firm. You are given two arrays, start and end each of size N.For an index 'i', start[i] denotes the starting time of the ith meeting while end[i] will denote the ending time of the ith meeting. Find the maximum number of meetings that can be accommodated if only one meeting can happen in the room at a particular time. Print the order in which these meetings will be performed.

Example: Input: N = 6, start[] = {1,3,0,5,8,5}, end[] = {2,4,5,7,9,9}
Output: 1 2 4 5
Explanation: See the figure for a better understanding.



Meeting No.	1 ✓	2 ✓	3	4 ✓	5 ✓	6
Start Time	1	3	0	5	8	5
End Time	2	4	5	7	9	9

Minimum number of platforms required for a railway

Problem Statement: We are given two arrays that represent the arrival and departure times of trains that stop at the platform. We need to find the minimum number of platforms needed at the railway station so that no train has to wait.

Examples 1:

```
Input: N=6,
arr[] = {9:00, 9:45, 9:55, 11:00, 15:00, 18:00}
dep[] = {9:20, 12:00, 11:30, 11:50, 19:00, 20:00}
Output:3
```

Job Sequencing Problem

Problem Statement: You are given a set of N jobs where each job comes with a **deadline** and **profit**. The profit can only be earned upon completing the job within its deadline. Find the **number of jobs** done and the **maximum profit** that can be obtained. Each job takes a **single unit** of time and only **one job** can be performed at a time.

Examples

Example 1: Input: N = 4,
Jobs = {(1,4,20),(2,1,10),(3,1,40),(4,1,30)}
Output: 2 60
Explanation: The 3rd job with a deadline 1 is performed during the first unit of time .The 1st job is performed during the second unit of time as its deadline is 4. Profit = 40 + 20 = 60

Example 2: Input: N = 5,
Jobs = {(1,2,100),(2,1,19),(3,2,27),(4,1,25),(5,1,15)}
Output: 2 127
Explanation: The first and third job both having a deadline 2 give the highest profit. Profit = 100 + 27 = 127

Fractional Knapsack Problem : Greedy Approach

Problem Statement: The weight of **N** items and their corresponding values are given. We have to put these items in a knapsack of weight **W** such that the **total value** obtained is **maximized.**

Note: We can either take the item as a whole or break it into smaller units.

Example:

Input: N = 3, W = 50, values[] = {100,60,120}, weight[] = {20,10,30}.

Output: 240.00

Explanation: The first and second items are taken as a whole while only 20 units of the third item is taken. Total value = 100 + 60 + 80 = 240.00

Find minimum number of coins

Problem Statement: Given a value V, if we want to make a change for V Rs, and we have an infinite supply of each of the denominations in Indian currency, i.e., we have an infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.

Examples:

Example 1:
Input: V = 70
Output: 2
Explaination: We need a 50 Rs note and a 20 Rs note.

Example 2:
Input: V = 121
Output: 3
Explaination: We need a 100 Rs note, a 20 Rs note and a 1 Rs coin.

Chocolate Distribution Problem

- Given an array of n integers where each value represents the number of chocolates in a packet. Each packet can have a variable number of chocolates. There are m students, the task is to distribute chocolate packets such that:

1. Each student gets one packet.
2. The difference between the number of chocolates in the packet with maximum chocolates and packet with minimum chocolates given to the students is minimum.

- ***Input :*** *arr[] = {7, 3, 2, 4, 9, 12, 56} , m = 3*
  Output: *Minimum Difference is 2*
  Explanation:
  We have seven packets of chocolates and we need to pick three packets for 3 students
  If we pick 2, 3 and 4, we get the minimum difference between maximum and minimum packet sizes.
- ***Input :*** *arr[] = {3, 4, 1, 9, 56, 7, 9, 12} , m = 5*
  Output: *Minimum Difference is 6*
  Explanation:
  The set goes like 3,4,7,9,9 and the output is 9-3 = 6
- ***Input :*** *arr[] = {12, 4, 7, 9, 2, 23, 25, 41, 30, 40, 28, 42, 30, 44, 48, 43, 50} , m = 7*
  Output: *Minimum Difference is 10*
  Explanation:
  We need to pick 7 packets. We pick 40, 41, 42, 44, 48, 43 and 50 to minimize difference between maximum and minimum.

Maximize Toys

- Given an array **arr[ ]** of length **N** consisting cost of **N** toys and an integer **K** depicting the amount with you. Your task is to find maximum number of toys you can buy with **K** amount.

Example 1:
Input: N = 7 K = 50 arr[] = {1, 12, 5, 111, 200, 1000, 10}
Output: 4
Explaination: The costs of the toys you can buy are 1, 12, 5 and 10.

Example 2:
Input: N = 3 K = 100 arr[] = {20, 30, 50}
Output: 3
Explaination: You can buy all toys.

Largest number with given sum

- Geek lost the password of his super locker. He remembers the number of digits **N** as well as the sum **S** of all the digits of his password. He know that his password is the largest number of **N** digits that can be made with given sum **S**. As he is busy doing his homework, help him retrieving his password.

Input: N = 5, S = 12
Output: 93000
Explanation: Sum of elements is 12. Largest possible 5 digit number is 93000 with sum 12.

Example 2:
Input: N = 3, S = 29
Output: -1
Explanation: There is no such three digit number whose sum is 29.