GREEDY ALGORITHM

INTRODUCTION TO ALOGRITHM

Made by -
Shivam Joshi

INTRODUCTION

In this presentation, we are going to discuss my problem statement, how Greedy algorithm is helpful and how we can solve the problem with the greedy algorithm approach.

A **greedy algorithm** is a simple, intuitive algorithm that is used in optimization problems. The algorithm makes the optimal choice at each step as it attempts to find the overall optimal way to solve the entire problem

Condition for greedy

If both of the properties below are true, a greedy algorithm can be used to solve the problem.

- **Greedy choice property:** A global (overall) optimal solution can be reached by choosing the optimal choice at each step.
- **Optimal substructure:** A problem has an optimal substructure if an optimal solution to the entire problem contains the optimal solutions to the sub-problems.

Problem statement

Activity selection problem

You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time.

Approach

The greedy choice is to always pick the next activity whose finish time is least among the remaining activities and the start time is more than or equal to the finish time of previously selected activity. We can sort the activities according to their finishing time so that we always consider the next activity as minimum finishing time activity.

1) Sort the activities according to their finishing time

2) Select the first activity from the sorted array and print it.

3) Do following for remaining activities in the sorted array.

a) If the start time of this activity is greater than or equal to the finish time of previously selected activity then select this activity and print it.

Initially :

arr[] = { { 5,9 } , { 1,2 } , { 3,4 } , { 0,6 } , { 5,7 } , { 8,9 } }

Step 1:

Sort the array according to finish time
arr[] = { { 1,2 } , { 3,4 } , { 0,6 } , { 5,7 } , { 8,9 } , { 5,9 } }

Step 2:

Print first activity and make i = 0
print = ({ 1,2 })

Step 3:

arr[] = { { 1,2 } , { 3,4 } , { 0,6 } , { 5,7 } , { 8,9 } , { 5,9 } }
↑
j

Start[j] >= finish[i]. print({ 3,4 })
make i = j , j++

Step 4:

arr[] = { { 1,2 } , { 3,4 } , { 0,6 } , { 5,7 } , { 8,9 } , { 5,9 } }
↑
j

Start[j] < finish[i]. j++

Step 5:

arr[] = { { 1,2 } , { 3,4 } , { 0,6 } , { 5,7 } , { 8,9 } , { 5,9 } }
↑
j

Start[j] >= finish[i]. print ({ 5,7 })
make i = j , j++

Step 6:

arr[] = { { 1,2 } , { 3,4 } , { 0,6 } , { 5,7 } , { 8,9 } , { 5,9 } }
↑
j

make i = j , j++

Step 6:

arr[] = { { 1,2 } , { 3,4 } , { 0,6 } , { 5,7 } , { 8,9 } , { 5,9 } }
Start[j] < finish[i].