CHAPTER FOUR
Greedy Algorithms

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment.

Greed Algorithm always makes the choice that looks best at the moment.

Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. So the problems where choosing locally optimal also leads to global solution are the best fit for Greedy.

A **Greedy algorithm** is an approach to solving a problem that selects the most appropriate option based on the current situation. This algorithm ignores the fact that the current best result may not bring about the overall optimal result. Even if the initial decision was incorrect, the algorithm never reverses it.

Greedy algorithms are algorithms that take the best, immediate, or local, solution while looking for an answer. They identify the globally (overall) optimal solution for certain optimization problems but might identify less-than-optimal solutions for certain instances of other problems. It follows the problem-solving heuristic of making the locally optimal choice at every stage.

Greedy Algorithms are simple, easy to implement and intuitive algorithms used in optimization problems. Greedy algorithms operate on the principle that if we continue making the locally optimal choice in each sub-problem we will form the global optimal choice for the entire problem.

Components of Greedy Algorithm

The components that can be used in the greedy algorithm are:

- **Candidate set:** A solution that is created from the set is known as a candidate set.
- **Selection function:** This function is used to choose the candidate or subset which can be added in the solution.
- **Feasibility function:** A function that is used to determine whether the candidate or subset can be used to contribute to the solution or not.
- **Objective function:** A function is used to assign the value to the solution or the partial solution.
- **Solution function:** This function is used to intimate whether the complete function has been reached or not.

Activity selection problem

1. Sort the activities as per finishing time in ascending order.
2. Select the first activity
3. Selection of the new activity if its starting time is greater than or equal to the previously selected activity's finish time.
4. Repeat step three (3) till all activates are checked.

Consider the following 8 activities
with their starting and finishing time

Activity	a1	a2	a3	a4	a5	a6	a7	a8
start	1	0	1	4	2	5	3	4
finish	3	4	2	6	9	8	5	5

our goal is to find non-conflicting activities

step 1: sort the activities as per finishing time in ascending order

Activity	a1	a2	a3	a4	a5	a6	a7	a8
start	1	0	1	4	2	5	3	4
finish	3	4	2	6	9	8	5	5

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

step 2: select the first activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

↑
i

Selected Activity	a3
start	1
finish	2

step 3: select next activity whose start time is greater than
or equal to the finish time of the previously selected activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

previously selected activity = i
next activity = j

↑ i ↑ j

start time of j = 1
finish time of i = 2

is
start time of j >= finish time of i
1 >= 2
NO
so, we move to next activity

Selected Activity	a3
start	1
finish	2

step 3: select next activity whose start time is greater than
or equal to the finish time of the previously selected activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

previously selected activity = i
next activity = j

↑ i ↑ j

start time of j = 0
finish time of i = 2

is
start time of j >= finish time of i
0 >= 2
NO
so, we move to next activity

Selected Activity	a3
start	1
finish	2

step 3: select next activity whose start time is greater than or equal to the finish time of the previously selected activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

previously selected activity = i
next activity = j

i (a3), j (a7)

start time of j = 3
finish time of i = 2

is
start time of j >= finish time of i
3 >= 2
YES
so, we select the activity j

now point i to newly selected activity

Selected Activity	a3	a7
start	1	3
finish	2	5

step 3: select next activity whose start time is greater than or equal to the finish time of the previously selected activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

previously selected activity = i
next activity = j

i j (a7)

start time of j = 3
finish time of i = 2

is
start time of j >= finish time of i
3 >= 2
YES
so, we select the activity j

now point i to newly selected activity

move to next activity

Selected Activity	a3	a7
start	1	3
finish	2	5

step 3: select next activity whose start time is greater than or equal to the finish time of the previously selected activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

previously selected activity = i
next activity = j

i (a7), j (a8)

start time of j = 4
finish time of i = 5

is
start time of j >= finish time of i
4 >= 5
NO
so, we move to next activity

Selected Activity	a3	a7
start	1	3
finish	2	5

step 3: select next activity whose start time is greater than or equal to the finish time of the previously selected activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

previously selected activity = i
next activity = j

i (a7), j (a4)

start time of j = 4
finish time of i = 5

is
start time of j >= finish time of i
4 >= 5
NO
so, we move to next activity

Selected Activity	a3	a7
start	1	3
finish	2	5

step 3: select next activity whose start time is greater than or equal to the finish time of the previously selected activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

previously selected activity = i
next activity = j

i (a7), j (a6)

start time of j = 5
finish time of i = 5

is
start time of j >= finish time of i
5 >= 5
YES
so, we select activity j

Selected Activity	a3	a7	a6
start	1	3	5
finish	2	5	8

step 3: select next activity whose start time is greater than or equal to the finish time of the previously selected activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

previously selected activity = i
next activity = j

i j (a6)

start time of j = 5
finish time of i = 5

is
start time of j >= finish time of i
5 >= 5
YES
so, we select activity j

now set i to the newly selected activity

Selected Activity	a3	a7	a6
start	1	3	5
finish	2	5	8

step 3: select next activity whose start time is greater than or equal to the finish time of the previously selected activity

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

$\uparrow i$ (a6) $\uparrow j$ (a5)

previously selected activity = i
next activity = j

start time of j = 2
finish time of i = 8

is
start time of j >= finish time of
2 >= 8
NO

Selected Activity	a3	a7	a6
start	1	3	5
finish	2	5	8

as we have reached the last activity so we stop here

Sorted Activity	a3	a1	a2	a7	a8	a4	a6	a5
start	1	1	0	3	4	4	5	2
finish	2	3	4	5	5	6	8	9

we have the required activity

Selected Activity	a3	a7	a6
start	1	3	5
finish	2	5	8