Experiment- 5

Problem Statement

Activiy Selection

Solution

The Activity Selection algorithm is a greedy algorithm used to solve the problem of selecting the maximum number of non-overlapping activities from a set of activities, where each activity has a start and end time. The algorithm works by selecting the activity with the earliest end time, and then selecting the next activity that does not overlap with the previously selected activity, and so on, until no more activities can be selected.

Algorithm:

GREEDY- ACTIVITY SELECTOR (s, f)

```
1. n ← length [s]
2. A ← {1}
3. j ← 1.
4. for i ← 2 to n
5. do if s_i ≥ f_i
6. then A ← A ∪ {i}
7. j ← i
8. return A
```

Theory:

The input to the algorithm is two lists: start_time and end_time, which represent the start and end times of the activities. The activities list is created by combining start_time and end_time into a list of tuples. The activities list is then sorted by the end time of each activity, using a lambda function to access the end time.

The algorithm then selects the first activity as the selected_activity list, and loops through the rest of the activities. If the start time of the current activity is greater than or equal to the end time of the last selected activity, the current activity is added to the selected_activity list.

At the end of the loop, the selected_activity list contains the maximum number of non-overlapping activities that can be selected.

we define the activity_selection function and then use it to select the maximum number of non-overlapping activities from two lists of start and end times. We then print the selected activities to the console.You can replace the start_time and end_time lists with your own data to test the function.

Code Implementation:

```
def activity_selection(start_time, end_time):
    # Combine start time and end time in a list
    activities = list(zip(start_time, end_time))

    # Sort activities by their end time
    activities.sort(key=lambda x: x[1])

    # Select the first activity
    selected_activity = [activities[0]]

    # Loop through the rest of the activities
    for activity in activities[1:]:
        # If the start time of the current activity is greater than or equal
to the end time of the last selected activity, select the current activity
        if activity[0] >= selected_activity[-1][1]:
            selected_activity.append(activity)

    return selected_activity

# Example usage
start_time = [1, 3, 0, 5, 8, 5]
end_time = [2, 4, 6, 7, 9, 9]
selected_activities = activity_selection(start_time, end_time)
print(selected_activities)
```

Output:

```
[(1, 2), (3, 4), (5, 7), (8, 9)]
```