Date ....

Expt. No. .... Page No. ....

ADA ASSIGNMENT - II

2K19/CO/819 RITIK SINGH

Activity Selection Problem

The Activity Selection Problem is an optimization problem which deals with the selection of non-conflicting activities that needs to be executed by a single person or machine in a given time frame.

Each activity is marked by a start and finish time. Greedy technique is used for finding the solution since this is an optimization problem.

* Steps for Activity Selection Problem

Following are the steps we will be following to solve the activity selection problem.

Step1: Sort the given activities in ascending order according to their finishing time.

Step2: Select the first activity from sorted array act[] and add it to sol[] array.

Step3: Repeat steps 4 and 5 for the remaining activities in act[].

Step4: If the start time of the currently selected activity is greater than or equal to the finish time of previously selected activity, then add it to the sol[] array

Step 5: Select the next activity in act[] array.

Step6: Print the sol[] array.

Teacher's Signature : ____________

* Activity Selection Problem Example

Let's try to trace the steps of above algorithm using an example:
In the table below, we have 6 activities with corresponding start and end time, the objective is to compute an execution schedule having maximum number of non-conflicting activities.

Start Time (s)	Finish Time (s)	Activity Name
5	9	a1
1	2	a2
3	4	a3
0	6	a4
5	7	a5
8	9	a6

Answer Step1: Sort the given activities in ascending order to their finishing time.
The table after we have sorted it:

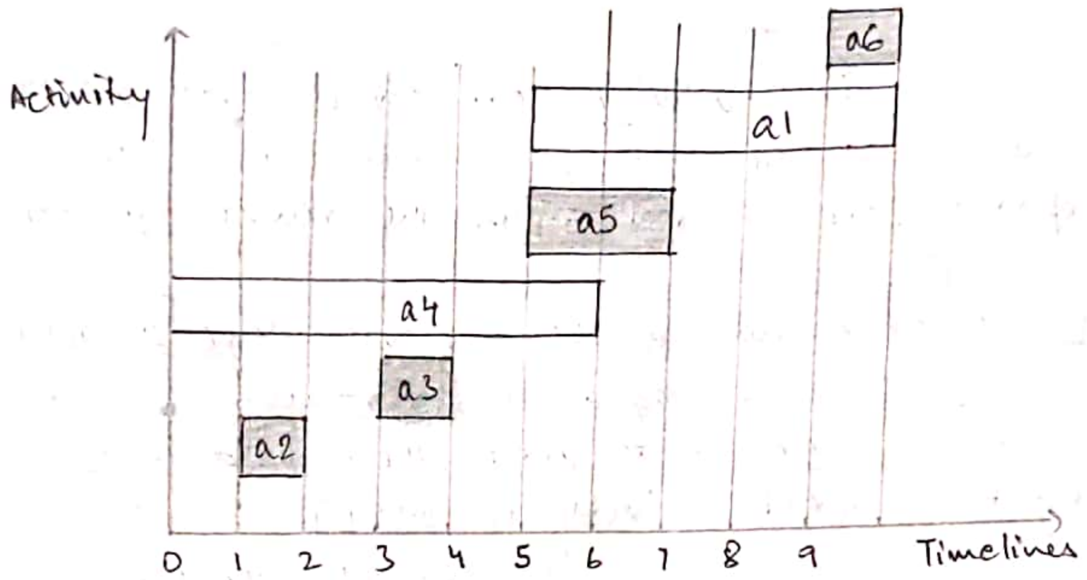
Start time (s)	Finish Time (f)	Activity Name
1	2	a2
3	4	a3
0	6	a4
5	7	a5
5	9	a1
8	9	a6

Scanned with CamScanner

Step2: Select the first activity from sorted array act[] and add it to the sol[] array, thus sol = {a2}.
Step3: Repeat the steps 4 and 5 for the remaining activities in act[].
Step4: If the start time of the currently selected activity is greater than or equal to the finish time of the previously selected activity, then add it to sol[].
Step5: Select the next activity in act[]
For the data given in the above table,
A. Select activity a3. Since the start time of a3 is greater than the finish time of a2. (i.e s(a3) > f(a2)) we add a3 to the solution set. Thus sol = {a2, a3}.
B. Select a4. Since s(a(4) < f(a3)), it is not added to the solution set.
C. Select a5. Since s (a5) > f(a3), a5 gets added to solution set. Thus sol = {a2, a3, a5}
D. Select a1. Since s(a1) < f (a5), a1 is not added to the solution set.
E. Selected a6. a6 is added to the solution set since s (a6) > f(a5). Thus sol = {a2, a3, a5, a6}.
Step 6: At last, print the array sol[]
Hence, the execution schedule of maximum of non-conflicting activities will be:

OUTPUT:
```
(1, 2)
(3,4)
(5,7)
(8,9)
```

Teacher's Signature : ________



In the above diagram, the selected activities have been highlighted in grey.

Time Complexity Analysis

Following are the scenarios for computing the time complexity of Activity Selection Algorithm.

- Case 1: When a given set of activities are already sorted according to their finishing time, then there is no sorting mechanism involved, in such a case the complexity of the algorithm will be $O(n)$.

- Case 2: When a given set of activities is unsorted, then we will have to use the sort() method defined in bits/stdc++ header file for sorting the activities list. The time complexity of this method will be $O(n \log n)$, which is also defines complexity of the algorithm.

* Real-Life Applications of Activity Selection Problem

Following are some of the real-life applications of this problem:

- Scheduling multiple competing events in a room, such that each event has its own start and end time.
- Scheduling manufacturing of multiple products on the same machine, such that each product has its own production timelines.
- Activity selection is one of the most well-known generic problems used in Operations Research for dealing with real life business problems.

Teacher's Signature : ____________