Activity Selection Problem

Given **N** activities with their start and finish day given in array **start[ ]** and **end[ ]**. 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 given day.
Note : Duration of the activity includes both starting and ending day.

Example 1:

Input:
N = 2
start[] = {2, 1}
end[] = {2, 2}
Output:
1
Explanation:
A person can perform only one of the given activities.

Example 2:

Input:
N = 4
start[] = {1, 3, 2, 5}
end[] = {2, 4, 3, 6}
Output:
3
Explanation:
A person can perform activities 1, 2 and 4.

Activity Selection Problem

The activity selection problem is a mathematical optimization problem. Our first illustration is the problem of scheduling a resource among several challenge activities. We find a greedy algorithm provides a well designed and simple method for selecting a maximum- size set of manually compatible activities.

Suppose S = {1, 2....n} is the set of n proposed activities. The activities share resources which can be used by only one activity at a time, e.g., Tennis Court, Lecture Hall, etc. Each Activity "i" has **start time** s_i and a **finish time** f_i, where $s_i \leq f_i$. If selected activity "i" take place meanwhile the half-open time interval $[s_i,f_i)$. Activities i and j are **compatible** if the intervals (s_i, f_i) and $[s_i, f_i)$ do not overlap (i.e. i and j are compatible if $s_i \geq f_i$ or $s_i \geq f_i$). The activity-selection problem chosen the maximum- size set of mutually consistent activities.

Algorithm Of Greedy- Activity Selector:

GREEDY- ACTIVITY SELECTOR (s, f)
1. n ← length [s]
2. A ← {1}
3. j ← 1.
4. for i ← 2 to n
5. do if $s_i \geq f_i$
6. then A ← A ∪ {i}
7. j ← i
8. return A

Example: Given 10 activities along with their start and end time as

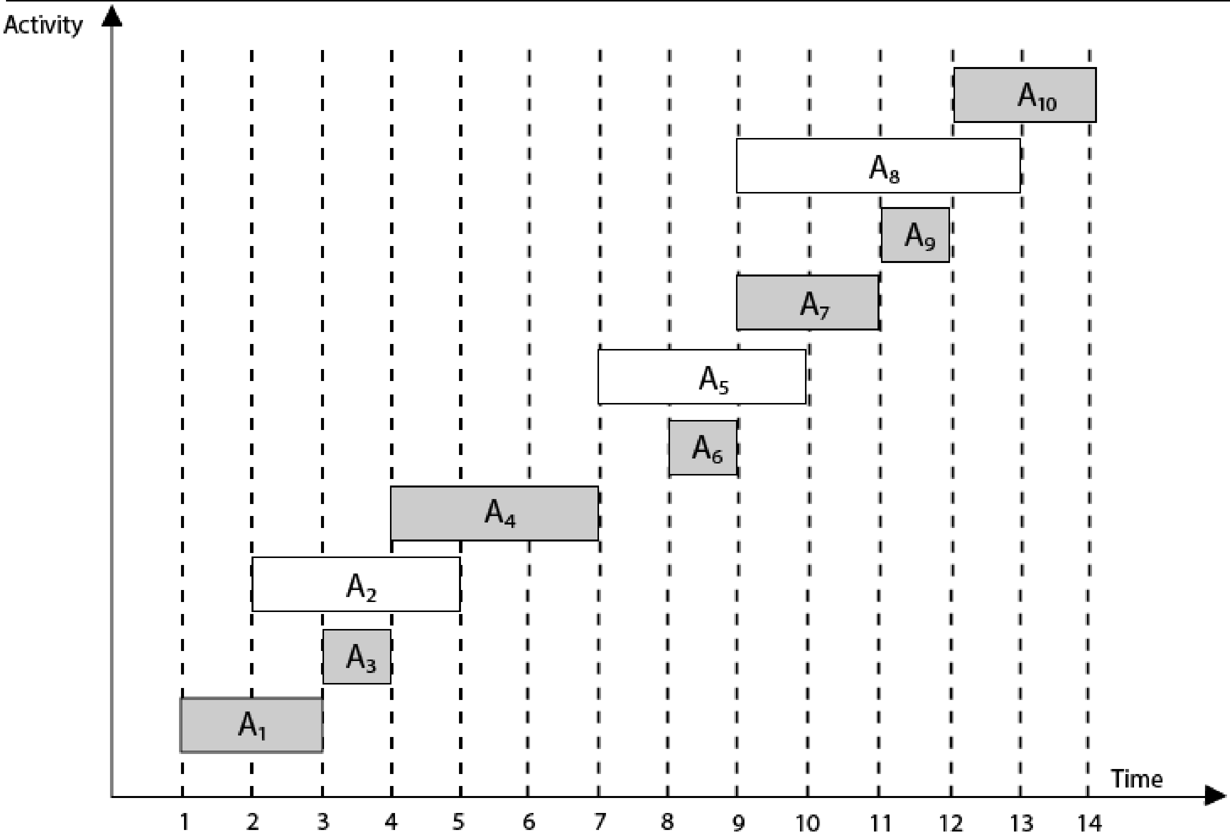
S = (A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A9 A10)
Si = (1,2,3,4,7,8,9,9,11,12)
fi = (3,5,4,7,10,9,11,13,12,14)

Compute a schedule where the greatest number of activities takes place.

Solution: The solution to the above Activity scheduling problem using a greedy strategy is illustrated below:

Arranging the activities in increasing order of end time

Activity	A_1	A_3	A_2	A_4	A_6	A_5	A_7	A_9	A_8	A_{10}
Start	1	3	2	4	8	7	9	11	9	12
Finish	3	4	5	7	9	10	11	12	13	14



Now, schedule A_1

Next schedule A_3 as A_1 and A_3 are non-interfering.

Next **skip** A_2 as it is interfering.

Next, schedule A_4 as A_1 A_3 and A_4 are non-interfering, then next, schedule A_6 as A_1 A_3 A_4 and A_6 are non-interfering.

Skip A_5 as it is interfering.

Next, schedule A_7 as A_1 A_3 A_4 A_6 and A_7 are non-interfering.

Next, schedule A_9 as A_1 A_3 A_4 A_6 A_7 and A_9 are non-interfering.

Skip A_8 as it is interfering.

Next, schedule A_{10} as A_1 A_3 A_4 A_6 A_7 A_9 and A_{10} are non-interfering.

Thus the final Activity schedule is:

$\boxed{(\mathbf{A_1\ A_3\ A_4\ A_6\ A_7\ A_9\ A_{10}})}$