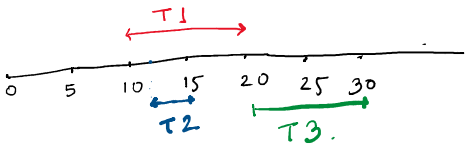
Activity selection problem

12 March 2025 15:24

Activity selection problem / maximum disjoint interval

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.

Task	Start Time	End Time
1	10	20
2	12	15
3	20	30



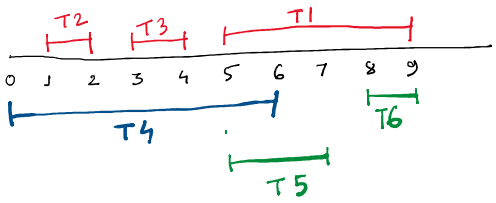
Possible activities that can be performed either

Ans- T1, T3 → 2

T2, T3 → 2

Eg-

Start Time (s)	Finish Time (f)	Task Name
5	9	T1
1	2	T2
3	4	T3
0	6	T4
5	7	T5
8	9	T6



T4, T6 , T2, T3, T1

Ans - 4

Possible ordering -

T2, T3, T5, T6 → 4.

Algorithm:

1. Sort all activities based on their finish time.
2. Choosing the first activity from the sorted list.
3. Select the next activity from the sorted list only if its `start` time is greater than or equal to the `finish` time of the previously selected activity.
4. Repeat `Step 3` for all the remaining activities in the sorted list.

Question: Maximum tasks that can be performed without any overlapping

Start Time (s)	Finish Time (f)	Task Name
5	9	T1
1	2	T2
3	4	T3
0	6	T4
5	7	T5
8	9	T6

Input -

n = 3

start[] = {10, 12, 20}

end[] = {20, 15, 30}.

Output - 2

Answer:

1. Sort all activities based on their finish time.

Start Time (s)	Finish Time (f)	Task Name
1	2	T2 ✓
3	4	T3 ✓
0	6	T4 ✗
5	7	T5 ✓
5	9	T1 ✗
8	9	T6 ✓

T2 → T3 → T5 → T6 → 4.

8-9 → DSA.

8-9 → DSA.

8:30 - 9:30 → ML.

Activities list →

Activities	Activity	Activity
10, 20	12, 15	20, 30

Input -

n = 3

start[] = {10, 12, 20}

end[] = {20, 15, 30}.

→ end = {15, 20, 30}.

Data structure:

start, end

S	E
10	20
12	15
20	30

→

S	E
10	15
12	20
20.	30

S	E
12	15
10	20
20	30

```
class Activity{
    int start;
    int end;
};
```

Integer [] arr =

n → no of activities.

~~Arraylist~~

Activity [] activity =

activity[i].start time

activity[i].end time.

Collections,

Comparator;

Java →

```
Collections.sort(activities, new ActivityComparator());
```

```
Collections.sort(activities);
```

↓

Sort based on their start time.

```
return a1.end - a2.end;
```

→ -ive → a1, a2

→ 0 → order Don't Matter

→ +ive → a2, a1.