25 march z section

25 March 2025 08:20

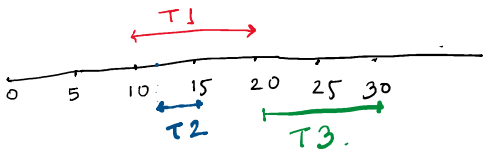
8 – 9 → DSA

8:30 – 9:30 → ML

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



Ans – T1, T3 → 2 ; T2, T3 → 2

Possible activities that can be performed either

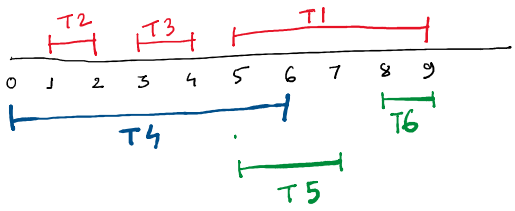
start = {10, 12, 20}

end = {20, 15, 30}

end' = {15, 20, 30}

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



Possible or during –

T2, T3, T1 ← 3 tasks

T2, T3, T5, T6 ← 4 tasks.

T4, T6 ← 2 tasks.

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

① Sort

② 1st task को हमेशा Perform Karo.

③ previous task.end time <= next task.start time.

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}

Code –

8 9 6

Code-

start[ ] = {s1, s2, s3
end[ ] = {e1, e2, e3}.

}

no of activities = start.length();

n = No. of tasks
0 1 2
start = {10, 12, 20}
end = {20, 15, 30}.

```
class Task{
    int start_time;
    int end_time;
};
```

Task t[3];

t[0].start_time
t[0].end_time.

Integer a[ ];

```
ArrayList<Task> task = new ArrayList<>();
for(i=0; i<n; i++)
{
    task.add(new Task(start[i], end[i]));
}
```

task.get(1).start_time;
task.get(1).end_time;