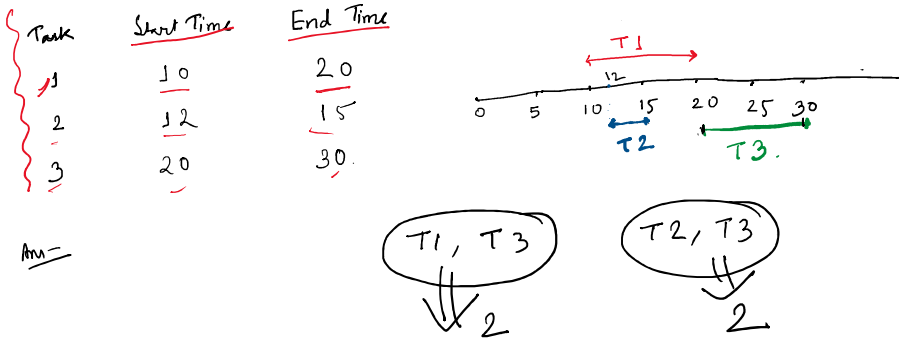
Activity selection problem

12 March 2025 15:24

Activity selection problem / maximum disjoint interval

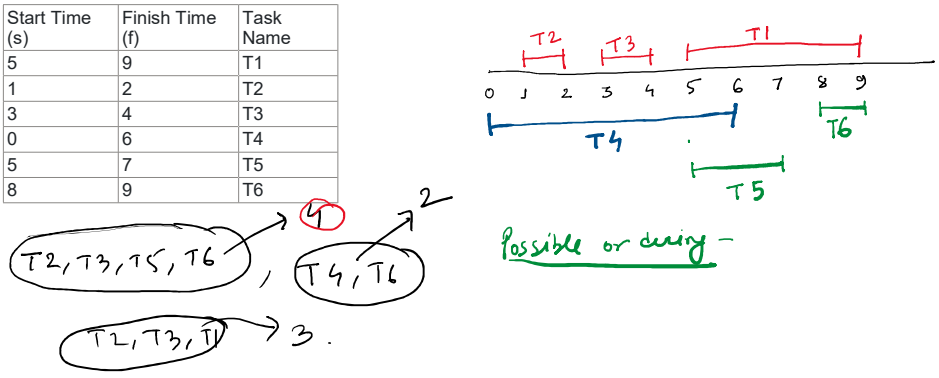
8 – 9 – DSA
8:30 – 9:30 – ML

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.



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



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

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 activities.

IP → n = No of task = 3.
start[] = {10, 12, 20}
end[] = {20, 15, 30}.
O/P → 2.

previous_task.end time <= next task. starting time.

Answer:

1. Sort all activities based on their finish time.

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

IP → n = No of task = 3.

→ start[] = {10, 12, 20}

→ end[] = {20, 15, 30}

O/P ⇒ 2.

Data structure / Container.

```
class Activity {
    int start_time;
    int end_time;
    //Constructor
}
```

```
ArrayList<Activity> activities = []
for(i=0; i<3; i++) {
    activities.add(new Activity(start time, end time))
}
```

Activity a1 = new Activity();

a1.start time

a1.end time

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

Collections.sort(activities) ⇒ Activities sort based on their start time.

↳ List.

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

-ive

a1.end < a2.end

a1

a2

0

a1.end = a2.end

a1 or a2

a2 a1

+ive.

a1.end > a2.end.

a2

a1

H/W — Java. Comparator

H/W: Job Sequencing Problem

Gfg

Fractional Knapsack Problem