

Design And Analysis Algorithms

Practical

Objective:- Implement and analyze the complexity of Selection Sort.

Code:-

```
import java.util.ArrayList;
import java.util.Collections;
class ActivitySelection {
    static class Activity {
        int start;
        int end;
        public Activity(int start, int end) {
            this.start = start;
            this.end = end;
        }
        @Override
        public String toString(){
            return "[" + this.start + ", " + this.end + "]";
        }
    }
    public static void maxActivities(ArrayList<Activity> activities){
        System.out.println("Given Activities: " + activities);
        Collections.sort(activities, (o1, o2) -> o1.end - o2.end);
        ArrayList<Activity> selectedActivities = new ArrayList<>();
        int currentEnd = -1;
        for (int i = 0; i < activities.size() ; i++) {
            Activity currentActivity = activities.get(i);
            if(currentActivity.start > currentEnd){
                selectedActivities.add(currentActivity);
                currentEnd = currentActivity.end;
            }
        }
        System.out.println("Selected Activities: " + selectedActivities);
    }
    public static void main(String[] args) {
        ArrayList<Activity> activities = new ArrayList<>();
        activities.add(new Activity(2, 5));
        activities.add(new Activity(3, 6));
        activities.add(new Activity(9, 12));
        activities.add(new Activity(11, 15));
        activities.add(new Activity(14, 16));
        activities.add(new Activity(17, 22));
        maxActivities(activities);
    }
}
```

Output:-

The screenshot shows an IDE with a project named 'DAA_Lab'. The 'src' folder contains several files, including 'CountingSort.java', 'FractionalKnapSack.java', 'LinearSearch.java', 'MergeSort.java', and 'SelectionSort.java'. The 'Run' window displays the output of the 'ActivitySelection' class, which prints the given activities and the selected activities.

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 class ActivitySelection {
4     static class Activity {
5         int start;
6         int end;
7         public Activity(int start, int end) {
8             this.start = start;
9             this.end = end;
10        }
11        @Override
12        public String toString(){
13            return "[" + this.start + ", " + this.end + "]";
14        }
15    }
16    public static void maxActivities(ArrayList<Activity> activities){
17        System.out.println("Given Activities: " + activities);
18        Collections.sort(activities, (o1, o2) -> o1.end - o2.end);
19        ArrayList<Activity> selectedActivities = new ArrayList<>();
20        int currentEnd = -1;
```

Run: ActivitySelection

C:\Users\lenovo\.jdk\openjdk-15.0.2\bin\java.exe ...

Given Activities: [[2, 5], [3, 6], [9, 12], [11, 15], [14, 16], [17, 22]]

Selected Activities: [[2, 5], [9, 12], [14, 16], [17, 22]]

Process finished with exit code 0

Analyze:-

The complexity of this problem is $O(n \log n)$ when the list is not sorted. When the sorted list is provided the complexity will be $O(n)$.