**PRACTICAL NO. 3b**

Roll no.: 14

Batch: B1

Name: Abhishek Angadi

Aim: Implement activity selection algorithm for the given scenario. Problem Definition: In the single-machine scheduling problem, we are given a set of n activities Ai. Each job i has a starting time si, deadline di and profit pi. At any time instant, we can do only one job. Doing a job i earns a profit pi. Generate a solution to select the largest set of mutually compatible jobs and calculate the total profit generated by the machine. The greedy algorithm for single-machine scheduling selects the job using activity selection algorithm.

Code:

public class PracticalB {

private static class Activity {

double start;

double finish;

String ac\_name;

double profit;

Activity(double start, double finish, String ac\_name, double profit) {

this.start = start;

this.finish = finish;

this.ac\_name = ac\_name;

this.profit = profit;

}

}

public static void bubbleSort(Activity[] activities) {

int n = activities.length;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - 1 - i; j++) {

if (activities[j].finish > activities[j + 1].finish) {

Activity temp = activities[j];

activities[j] = activities[j + 1];

activities[j + 1] = temp;

}

}

}

}

public static void Act\_selection(Activity[] activities) {

bubbleSort(activities);

System.out.println("Selected Activities:");

double totalProfit = 0;

System.out.println("Activity: " + activities[0].ac\_name + ", Start: " + activities[0].start + ", Finish: " + activities[0].finish + ", Profit: " + activities[0].profit);

totalProfit += activities[0].profit;

double lastFinishTime = activities[0].finish;

for (int i = 1; i < activities.length; i++) {

if (activities[i].start >= lastFinishTime) {

System.out.println("Activity: " + activities[i].ac\_name + ", Start: " + activities[i].start + ", Finish: " + activities[i].finish + ", Profit: " + activities[i].profit);

totalProfit += activities[i].profit;

lastFinishTime = activities[i].finish;

}

}

System.out.println("Final Profit: " + totalProfit);

}

public static void main(String[] args) {

Activity[] activities = {

new Activity(1, 4, "A1", 10),

new Activity(3, 5, "A2", 15),

new Activity(0, 6, "A3", 14),

new Activity(5, 7, "A4", 12),

new Activity(3, 9, "A5", 20),

new Activity(5, 9, "A6", 30),

new Activity(6, 10, "A7", 32),

new Activity(8, 11, "A8", 28),

new Activity(8, 12, "A9", 30),

new Activity(2, 14, "A10", 40),

new Activity(12, 16, "A11", 45)

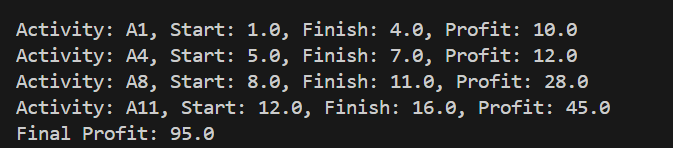
};

Act\_selection(activities);

}

}

Output:



Conclusion:

Hence, we successfully implemented activity selection algorithm for the given scenario.

Github: https://github.com/Shadow3456rh/