G. H. RAISONI COLLEGE OF ENGG., NAGPUR
(An Autonomous Institute under UGC Act 1956)
Department of Computer Science & Engg.

Date: 04/08/2020

Practical Subject: Design and Analysis of Algorithms
Session: 2020-21

Student Details:

Roll Number	58
Name	Shivam Tawari
Semester	3
Section	A
Branch	Artificial Intelligence

Practical Details: Practical Number- 4

Practical Aim	To Implement and Analyze time complexity of Greedy Algorithm
Theory & Algorithm	***Greedy Algorithm:*** Greedy algorithm is designed to achieve optimum solution for a given problem. In greedy algorithm approach, decisions are made from the given solution domain. As being greedy, the closest solution that seems to provide an optimum solution is chosen. Greedy algorithms try to find a localized optimum solution, which may eventually lead to globally optimized solutions. However, generally greedy algorithms do not provide globally optimized solutions. ***Activity Selection:*** The activity selection problem is a combinatorial optimization problem concerning the selection of non-conflicting activities to perform within a given time frame, given a set of activities each marked by a start time (s_i) and finish time (f_i). The problem is to select the maximum number of activities that can be performed by

	a single person or machine, assuming that a person can only work on a single activity at a time. The activity selection problem is also known as the Interval scheduling maximization problem (ISMP), which is a special type of the more general Interval Scheduling problem. ***Activity Selection Algorithm:*** Step 1: START Step 2: Input Activity start and finish time Step 3: Sort Activities in ascending order according to finish time Step 4: Select the first activity and make i=0, j=1 Step 5: If start[j] >= finish[i], select the activity and make i=j Step 6: Increment j by 1 Step 7: If j < total activities, go to Step 5 else Step 8 Step 8: Print Selected Activities Step 9: STOP
Complexity	***Worst Case:*** O (n logn) ***Best Case:*** Ω (n)

Program

main.cpp

Run

```cpp
#include "iostream"
#include "algorithm"

using namespace std;

struct Activity
{
    int start, finish;
};

bool activityCompare(Activity s1, Activity s2)
{
    return (s1.finish < s2.finish);
}

void printMaxActivities(Activity arr[], int n)
{
    sort(arr, arr+n, activityCompare);
    Activity scheduled[10];
    int i = 0, k = 1, count = 1;
    scheduled[0] = arr[0];
    for (int j = 1; j < n; j++)    {
        if (arr[j].start >= arr[i].finish)  {
            count++;
            scheduled[k] = arr[j];
            k++;
```

```
            i = j;
        }
    }
    cout << "\n Maximum " << count << " activities can be scheduled."
        ;
    cout << "\n Following activities are selected: ";
    for (i = 0; i<count; i++)   {
        cout << "(" << scheduled[i].start << ", "
            << scheduled[i].finish << "), ";
    }
}

int main()
{
    Activity act[10];
    int n;
    cout << "\n Name: Shivam Tawari";
    cout << "\nSection: A";
    cout << "\nRoll Number: 58";
    cout << "\nEnter Maximum number of Activities: ";
    cin >> n;
    for (int i=0; i<n; i++) {
        cout << "\n Enter Activity " << i+1 << " Start and End Time:
            ";
        cin >> act[i].start >> act[i].finish;
    }
    printMaxActivities(act, n);
    return 0;
}
```

Output

Output Clear

```
g++ -o /tmp/UMeUKX8UHJ.o /tmp/UMeUKX8UHJ.cpp
/tmp/UMeUKX8UHJ.o
Name: Shivam Tawari
Section: A
Roll Number: 58
Enter Maximum number of Activities: 3
Enter Activity 1 Start and End Time: 4
2
Enter Activity 2 Start and End Time: 1
2
Enter Activity 3 Start and End Time: 3
7
Maximum 2 activities can be scheduled.
 Following activities are selected: (4, 2), (3, 7),
```