 Marwadi University	Marwari University Faculty of Technology Department of Information and Communication Technology	
Subject: Design and Analysis of Algorithms (01CT0512)	Aim: Implementing application-based algorithms using Greedy Approach	
Experiment No: 07	Date:	Enrollment No: 92200133030

Aim: Implementing application-based algorithms using a Greedy Approach

IDE: Visual Studio Code

I. Job Scheduling Problem

Theory: -

- The Job Scheduling Problem (JSP) is a classic optimization problem that involves scheduling a set of jobs on resources (like machines) in a way that optimizes a particular objective, such as minimizing total completion time, minimizing delays, or maximizing resource utilization. A greedy algorithm approach is a common and efficient way to address certain variants of JSP, as it focuses on building an optimal solution step-by-step by making a locally optimal choice at each step with the hope of reaching a globally optimal solution.

1. Problem Definition


- Given a set of n jobs, each with a specific processing time and possibly a deadline, the objective is to assign these jobs to resources or time slots in such a way that some objective function, f , is minimized or maximized. Some common objectives include:
 - Minimizing total completion time.
 - Minimizing the maximum completion time (makespan).
 - Minimizing delays or tardiness, where each job has a due date.

2. Greedy Approach to Job Scheduling

- The greedy approach to JSP is based on a strategy of prioritizing jobs according to a specific criterion and scheduling them in that order. This approach can provide optimal solutions for certain types of job scheduling problems, especially when the objective function aligns well with the chosen greedy criterion.

3. Greedy Criteria for Different Variants of JSP

- In job scheduling problems, different greedy criteria can be selected depending on the objective:
 - Shortest Processing Time First (SPT): For minimizing total completion time, a common greedy criterion is to select the job with the shortest processing time next. This heuristic works well for reducing average job completion time.
 - Earliest Deadline First (EDF): For minimizing tardiness or meeting deadlines, jobs are sorted by their due dates, and the job with the earliest due date is scheduled first. This strategy can be particularly effective when minimizing the number of late jobs.
 - Highest Profit First: In situations where jobs have associated profits or rewards, a greedy algorithm can prioritize jobs with the highest profit-to-processing-time ratio, scheduling them in order to maximize total profit.

 Marwadi University	Marwari University Faculty of Technology Department of Information and Communication Technology	
Subject: Design and Analysis of Algorithms (01CT0512)	Aim: Implementing application-based algorithms using Greedy Approach	
Experiment No: 07	Date:	Enrollment No: 92200133030

Programming Language: - C++


Code :-

```
#include <bits/stdc++.h>
using namespace std;

class Job {
public:
    int id;
    int deadline;
    int profit;
    Job(int id, int deadline, int profit): id(id), deadline(deadline), profit(profit) {}
};

pair<int, long long> Job_Scheduling(vector<Job> jobs) {
    long long Max_Profit = 0.0;
    int count_of_jobs = 0;
    int max_deadline = -1;
    sort(jobs.begin(), jobs.end(), [](const Job& a, const Job& b) {
        return a.profit > b.profit;
    });
    for (const Job& job : jobs) {
        max_deadline = max(max_deadline, job.deadline);
    }
    vector<int> selectedJobs(max_deadline + 1, -1);
    for (int i = 0; i < jobs.size(); i++) {
        for (int j = jobs[i].deadline; j >= 0; j--) {
            if (selectedJobs[j] == -1) {
                selectedJobs[j] = jobs[i].id;
                count_of_jobs++;
                Max_Profit += jobs[i].profit;
                break;
            }
        }
    }

    return { count_of_jobs , Max_Profit };
}
```

 Marwadi University	Marwari University Faculty of Technology Department of Information and Communication Technology	
Subject: Design and Analysis of Algorithms (01CT0512)	Aim: Implementing application-based algorithms using Greedy Approach	
Experiment No: 07	Date:	Enrollment No: 92200133030

```

int main() {
    vector<Job> jobs;
    vector<int> deadline = { 2, 4, 6, 5, 2, 2, 6, 4 };
    vector<int> profit = { 22, 25, 20, 10, 65, 60, 70, 80 };

    for (int i = 0; i < 8; i++) {
        jobs.push_back(Job(i + 1, deadline[i], profit[i]));
    }

    pair<int, long long> Count_and_profit = Job_Scheduling(jobs);

    cout << "We can Perform " << Count_and_profit.first << " and Earn Profit of Rs. " <<
    Count_and_profit.second << " ." << endl;

    return 0;
}

```

Output :-

```

PS D:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 7> cd "d:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 7\" ; if ($?) { g++ Job_Scheduling.cpp -o Job_Scheduling } ; if ($?) { .\Job_Scheduling }
We can Perform 7 and Earn Profit of Rs. 342 .
PS D:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 7>

```


Space Complexity:- _____

Justification: -

Time Complexity:

Best Case Time Complexity: _____

Justification: -

 Marwadi University	Marwari University Faculty of Technology Department of Information and Communication Technology	
Subject: Design and Analysis of Algorithms (01CT0512)	Aim: Implementing application-based algorithms using Greedy Approach	
Experiment No: 07	Date:	Enrollment No: 92200133030

Worst Case Time Complexity:- _____

Justification: -


II. Activity Selection Problem :-

Theory: -

- The **Activity Selection Problem** is a classic optimization problem in computer science and operations research that aims to select the maximum number of non-overlapping activities from a given set, each with a start and end time. This problem is commonly solved using the **greedy approach**, which involves making a series of locally optimal choices to reach a global optimum.
- ❖ **Problem Definition**
- Given a set of activities, where each activity i has a **start time** s_i and an **end time** e_i , the objective is to select the maximum number of activities that do not overlap. In other words, if we choose an activity i , then no other selected activity j can have a start time s_j that is less than e_i .

Approach Using the Greedy Method

- The greedy method is particularly effective for this problem because it allows us to make an optimal choice at each step, without needing to look ahead or reconsider previous choices. The greedy approach for the Activity Selection Problem works as follows:
 1. **Sort the Activities by End Time:** The first step in the greedy approach is to sort all activities in ascending order based on their end times e_i . Sorting by end times helps ensure that the activity with the earliest finish time is considered first, maximizing the time remaining for other activities.
 2. **Select the Activity with the Earliest Finish Time:** After sorting, the first activity in the sorted list (the one that finishes earliest) is always selected, as it leaves the maximum room for other activities. We choose this activity and then discard any other activities that overlap with it.
 3. **Repeat the Process:** For each subsequent activity in the sorted list, check if its start time s_j is greater than or equal to the end time e_i of the previously selected activity. If it is, select this activity and update the end time to the finish time of this activity.
 4. **Continue Until All Activities Are Examined:** Repeat the above process until all activities have been considered.


 Marwadi University	Marwari University Faculty of Technology Department of Information and Communication Technology	
Subject: Design and Analysis of Algorithms (01CT0512)	Aim: Implementing application-based algorithms using Greedy Approach	
Experiment No: 07	Date:	Enrollment No: 92200133030

Code :-

```
#include <bits/stdc++.h>
using namespace std;
class Activity {
public:
    int id;
    int start;
    int end;
    Activity(int id , int start, int end) : id(id) , start(start), end(end) {};
};
vector<Activity> Activity_Selection(vector<Activity>& activities) {
    vector<Activity> selected_activities;
    sort(activities.begin(), activities.end(), [](Activity a, Activity b) {
        return a.end < b.end;
    });
    selected_activities.push_back(activities[0]);
    int last_activity_end = selected_activities.back().end;
    for (int i = 1; i < activities.size(); i++) {
        if (activities[i].start >= last_activity_end) {
            selected_activities.push_back(activities[i]);
            last_activity_end = selected_activities.back().end;
        }
    }
    return selected_activities;
}
int main() {

    vector<Activity> activities;
    vector<int> start_time = { 5,1,3,0,5,8 };
    vector<int> end_time = { 9,2,4,6,7,9 };
    for (int i = 0; i < start_time.size(); i++) {
        activities.push_back(Activity(i+ 1 ,start_time[i], end_time[i]));
    }
    vector<Activity> selected_activities = Activity_Selection(activities);
    cout << "Selected Activities are :- " << endl;
    for (auto activity : selected_activities) {
        cout << "Activity ID :- " << activity.id << " Start Time: " << activity.start << ", End
Time: " << activity.end << endl;
    }

    return 0;
}
```

 Marwadi University	Marwari University Faculty of Technology Department of Information and Communication Technology	
Subject: Design and Analysis of Algorithms (01CT0512)	Aim: Implementing application-based algorithms using Greedy Approach	
Experiment No: 07	Date:	Enrollment No: 92200133030

Output :-

```
PS D:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 7> cd "d:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 7\" ; if ($?) { g++ Activity_Selection.cpp -o Activity_Selection } ; if ($?) { .\Activity_Selection }
Selected Activities are :-
Activity ID :- 2 Start Time: 1, End Time: 2
Activity ID :- 3 Start Time: 3, End Time: 4
Activity ID :- 5 Start Time: 5, End Time: 7
Activity ID :- 6 Start Time: 8, End Time: 9
PS D:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 7> |
```

Space Complexity:- _____

Justification: -

Time Complexity:

Best Case Time Complexity: _____

Justification: -

Worst Case Time Complexity:- _____

Justification: -

Conclusion:-
