JOB SEQUENCING PROBLEM

DAA MINI-PROJECT

Team Members:

1. Shashank Nandanwar (RA2011003010779)
2. Adarsh Vardhan Singh (RA2011003010794)
3. Smarajit Baksi (RA2011003010813)

CONTENTS:

1. CONTRIBUTION TABLE
2. PROBLEM STATEMENT
3. PROBLEM EXPLANATION
4. DESIGN TECHNIQUE USED
5. ALGORITHM WORKING AND EXPLANATION
6. SOURCE CODE
7. COMPLEXITY ANALYSIS
8. CONCLUSION
9. REFERENCES

CONTRIBUTION TABLE

NAME	CONTRIBUTION
1. SMARAJIT BAKSI	Algorithm Design
2. ADARSH VARDHAN SINGH	Complexity Analysis
3. SHASHANK NANDANWAR	Source Code Generation

PROBLEM STATEMENT:

In job sequencing problem, the objective is to find a sequence of jobs, which is completed within their deadlines and gives maximum profit.

Example:

```
Input: Four Jobs with following
deadlines and profits
JobID   Deadline   Profit
  a        4          20
  b        1          10
  c        1          40
  d        1          30
Output: Following is maximum
profit sequence of jobs
         c, a
```

PROBLEM EXPLANATION:

The sequencing of jobs on a single processor with deadline constraints is called as Job Sequencing with Deadlines.

Here-

- You are given a set of jobs.
- Each job has a defined deadline and some profit associated with it.
- The profit of a job is given only when that job is completed within its deadline.
- Only one processor is available for processing all the jobs.
- Processor takes one unit of time to complete a job.

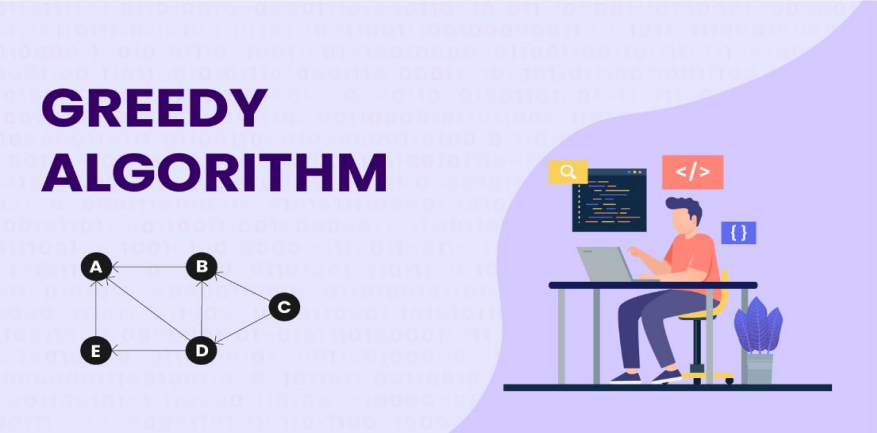
The problem states-

"How can the total profit be maximized if only one job can be completed at a time?"

DESIGN TECHNIQUE USED:

Greedy Algorithm is adopted to determine how the next job is selected for an optimal solution.

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.



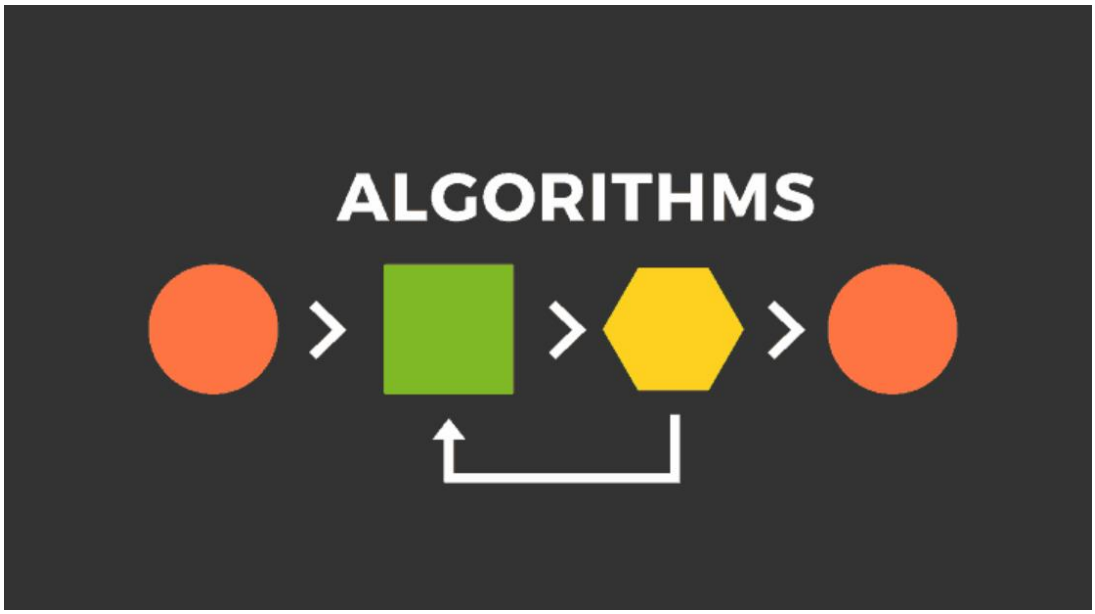
The greedy approach produces an optimal result in fairly less time. As each job takes the same amount of time, we can think of the schedule S consisting of a sequence of job slots 1, 2, 3, …, N, where S(t) indicates job scheduled in slot t. Slot t has a span of (t – 1) to t. S(t) = 0 implies no job is scheduled in slot t, where, schedule S is an array of slots S(t), S(t) ∈ {1, 2, 3, …, N} for each t ∈ {1, 2, 3, …, N}

ALGORITHM:

This is a standard Greedy Algorithm problem.

Following is the algorithm:-

1. Sort all jobs in descending order of profit.
2. Iterate on jobs in decreasing order of profit. For each job, do the following:
   a) Find a time slot i, such that slot is empty and i<deadline and i is the greatest. Put the job in this slot and mark this slot filled.
   b) If no such i exists, then ignore the job.



ALGORITHM EXPLANATION WITH EXAMPLE:-

Let us consider a set of given jobs as shown in the following table. We have to find a sequence of jobs, which will be completed within their deadlines and will give maximum profit. Each job is associated with a deadline and profit.

Job	**J_1**	**J_2**	**J_3**	**J_4**	**J_5**
Deadline	2	1	3	2	1
Profit	60	100	20	40	20

To solve this problem, the given jobs are sorted according to their profit in a descending order. Hence, after sorting, the jobs are ordered as shown in the following table.

Job	**J_2**	**J_1**	**J_4**	**J_3**	**J_5**
Deadline	1	2	2	3	1
Profit	100	60	40	20	20

From this set of jobs, first we select ***J_2***, as it can be completed within its deadline and contributes maximum profit.

- Next, ***J_1*** is selected as it gives more profit compared to ***J_4***.
- In the next clock, ***J_4*** cannot be selected as its deadline is over, hence ***J_3*** is selected as it executes within its deadline.
- The job ***J_5*** is discarded as it cannot be executed within its deadline.

Thus, the solution is the sequence of jobs (***J_2, J_1, J_3***), which are being executed within their deadline and gives maximum profit.

Total profit of this sequence is **100 + 60 + 20 = 180**.

SOURCE CODE:

```
#include <bits/stdc++.h>
using namespace std;
struct Job {
	char id;
	int dead;
	int profit;
};
struct jobProfit {
	bool operator()(Job const& a, Job const& b)
	{
		return (a.profit < b.profit);
	}
}
void printJobScheduling(Job arr[], int n)
{	vector<Job> result;
	sort(arr, arr + n,
		[](Job a, Job b) { return a.dead < b.dead; });
	priority_queue<Job, vector<Job>, jobProfit> pq;
	for (int i = n - 1; i >= 0; i--) {
		int slot_available;
		if (i == 0) {
			slot_available = arr[i].dead;
		}
		else {
```

```
                slot_available = arr[i].dead - arr[i - 1].dead;
                }
                pq.push(arr[i]);
                while (slot_available > 0 && pq.size() > 0) {
                        Job job = pq.top();
                        pq.pop();
                        slot_available--;
                        result.push_back(job);
                }
        }
        sort(result.begin(), result.end(),
                [&](Job a, Job b) { return a.dead < b.dead; });
        for (int i = 0; i < result.size(); i++)
                cout << result[i].id << ' ';
        cout << endl;
}
int main()
{       Job arr[] = { { 'a', 2, 100 },
                                { 'b', 1, 19 },
                                { 'c', 2, 27 },
                                { 'd', 1, 25 },
                                { 'e', 3, 15 } };
        int n = sizeof(arr) / sizeof(arr[0]);
        cout << "Following is maximum profit sequence of jobs "
                        "\n";
        printJobScheduling(arr, n);
        return 0; }
```

COMPLEXITY ANALYSIS:

Simple greedy algorithm spends most of the time looking for the latest slot a job can use.

On average, N jobs search N/2 slots. This would take $O(N^2)$ time.

Analysis:

1. Sort job according to decreasing order of deadline =O(nlogn)

2. For each job find slot in array of size n = O(n^2)

Therefore, Total time =O(nlogn) + O(n^2) =O(n^2)

CONCLUSION:

The Greedy Approach is used in the Job Sequencing problem as it produces the optimal solution in a fairly less amount of time. Thus, the Job Sequencing Algorithm was successfully designed and implemented using the Greedy Method in the mini-project.

REFERENCES:

- Algorithms Illuminated (Part 3): Greedy Algorithms and Dynamic Programming by Tim Roughgarden.

- https://www.gatevidyalay.com/job-sequencing-with-deadlines/

- https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_job_sequencing_with_deadline.html