K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: A1	**Roll No.: 16010123012**
Experiment No. 6	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Study, Implementation, and Analysis of Job Sequencing with Deadlines.

Objective: To learn the Greedy strategy of solving the problems for different types of problems

---

CO to be achieved:

CO 2	Describe various algorithm design strategies to solve different problems and analyse Complexity.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://lcm.csa.iisc.ernet.in/dsa/node184.htm**
4. **http://students.ceid.upatras.gr/~papagel/project/kruskal.htm**
5. **http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/kruskalAlgor.html**
6. **http://lcm.csa.iisc.ernet.in/dsa/node183.html**
7. **http://students.ceid.upatras.gr/~papagel/project/prim.htm**
8. **http://www.cse.ust.hk/~dekai/271/notes/L07/L07.pdf**

Pre Lab/ Prior Concepts:
Data structures, Concepts of algorithm analysis

Historical Profile: The Job Sequencing with Deadlines problem is a classic optimization problem where the goal is to schedule jobs to maximize profit, ensuring each job is completed before its deadline. The algorithm follows a greedy approach. The Job Sequencing with Deadlines Problem is a foundational optimization problem in scheduling theory and computer science. Its historical development is closely tied to advancements in algorithm design, combinatorial optimization, and practical applications in operations research and industrial processes.

Origins and Early Development

Scheduling Problems in Industry (Mid-20th Century): The problem emerged during the industrial revolution and later in the mid-20th century with the rise of operational research. Scheduling tasks to maximize efficiency, minimize delays, or maximize profits became critical for industries like manufacturing and logistics.

Development of Mathematical Formulation: Researchers began formalizing scheduling problems, including constraints like deadlines and profit maximization, laying the groundwork for job sequencing problems. The focus shifted from heuristic or ad-hoc methods to formal algorithmic solutions.

New Concepts to be learned:
Application of algorithmic design strategy to any problem, Greedy method of problem solving Vs other methods of problem solving, optimality of the solution, knapsack problem and their applications

Algorithm:

Input: A set of n jobs, where each job i has:

1. pi: Profit if the job is completed.
2. di: Deadline (maximum time slot by which it should be completed).

Output: A schedule of jobs such that:

1. Each job is completed within its deadline.
2. The profit is maximized.

Step 1: Sort Jobs by Profit. Sort all jobs in descending order of their profit pi. Jobs with higher profits are prioritized.

Step 2: Allocate Time Slots

1. Iterate through the sorted jobs.
2. For each job: Find the latest available time slot ≤ di (the deadline of the job).If a time slot is available, schedule the job in that slot and update the slot as occupied.

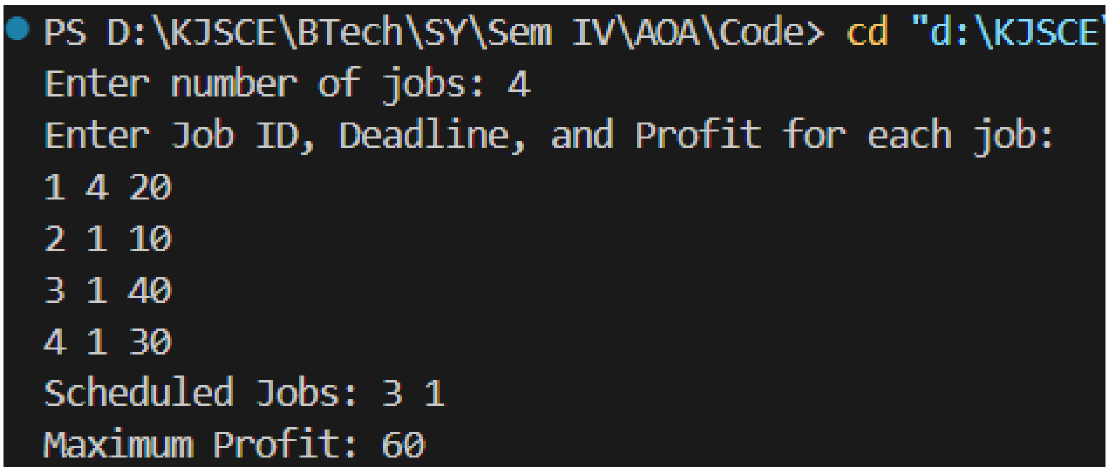
Step 3: Output the Job Schedule. (Return the scheduled jobs and the total profit.)

Code:

```cpp
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;

struct Job
{
  int id, deadline, profit;
};

bool compare(Job a, Job b) { return a.profit > b.profit; }

int job_scheduling(vector<Job> &jobs, int max_deadline)
{
  sort(jobs.begin(), jobs.end(), compare);
  vector<int> schedule(max_deadline, -1);
  int total_profit = 0, jobs_over = 0;

  for (Job &job : jobs)
  {
    for (int i = min(max_deadline, job.deadline) - 1; i >= 0; i--)
    {
      if (schedule[i] == -1)
      {
        schedule[i] = job.id;
        total_profit += job.profit;
        jobs_over++;
        break;
      }
    }
  }

  cout << "Scheduled Jobs: ";
  for (int i = 0; i < max_deadline; i++)
  {
    if (schedule[i] != -1)
    {
      cout << schedule[i] << " ";
    }
  }
  cout << endl;

  return total_profit;
}
```

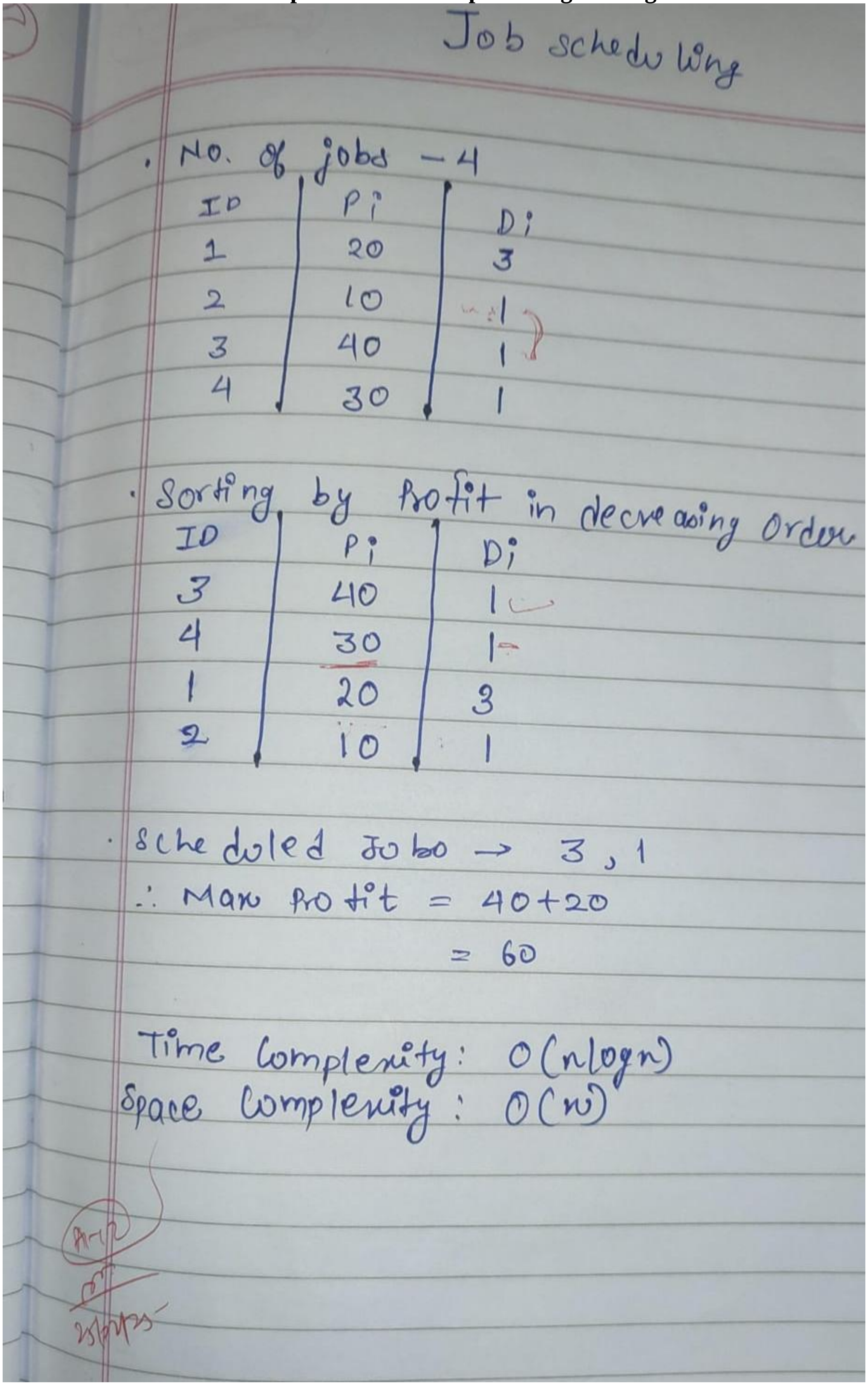
```cpp
int main()
{
  int n;
  cout << "Enter number of jobs: ";
  cin >> n;
  vector<Job> jobs(n);
  int max_deadline = 0;

  cout << "Enter Job ID, Deadline, and Profit for each job: " << endl;
  for (int i = 0; i < n; i++)
  {
    cin >> jobs[i].id >> jobs[i].deadline >> jobs[i].profit;
    max_deadline = max(max_deadline, jobs[i].deadline);
  }

  int max_profit = job_scheduling(jobs, max_deadline);
  cout << "Maximum Profit: " << max_profit << endl;
}
```

Output:



```
PS D:\KJSCE\BTech\SY\Sem IV\AOA\Code> cd "d:\KJSCE
Enter number of jobs: 4
Enter Job ID, Deadline, and Profit for each job:
1 4 20
2 1 10
3 1 40
4 1 30
Scheduled Jobs: 3 1
Maximum Profit: 60
```

Example / Analysis of algorithm:



Job scheduling

- No. of jobs – 4

ID	Pi	Di
1	20	3
2	10	1
3	40	1
4	30	1

- Sorting by Profit in decreasing Order

ID	Pi	Di
3	40	1
4	30	1
1	20	3
2	10	1

- Scheduled Jobs → 3, 1

∴ Max Profit = 40+20

= 60

Time Complexity: O(nlogn)
Space Complexity: O(n)

[illegible]

Conclusion:
I have completed the experiment, where I implemented and analyzed the Greedy Algorithm to maximize profit while ensuring jobs are scheduled within their deadlines. I observed that sorting jobs in descending order of profit allows for an optimal scheduling approach. By iterating backward from the latest available slot, I was able to efficiently allocate jobs while preserving earlier slots for jobs with stricter deadlines. I also analyzed the time complexity of the algorithm. The sorting step runs in $O(n \log n)$, while job placement takes at most $O(n \times d)$ in the worst case, where d is the maximum deadline.