Job Sequencing Problem

The Job Sequencing Problem is a classic optimization problem that can be solved using the greedy approach. The objective is to schedule jobs to maximize total profit, given deadlines and profits associated with each job. Here's an explanation of the problem and its solution:

Problem Statement

- You are given **n jobs**, each with the following attributes:
    - **Deadline (di):** The latest time by which the job must be completed.
    - **Profit (pi):** The profit earned if the job is completed within its deadline.
- Each job takes **1 unit of time**.
- Only one job can be scheduled at a time.

The goal is to schedule the jobs to maximize total profit, ensuring that no job is scheduled after its deadline.

Steps to Solve Using Greedy Approach

1. **Sort the Jobs by Profit (Descending Order):**
    - Start with the most profitable job to maximize total profit.
2. **Iterate Over Sorted Jobs:**
    - For each job, check if it can be scheduled before its deadline.
    - Use a timeline array (or similar structure) to track scheduled slots.
3. **Assign Jobs to Available Slots:**
    - Starting from the job's deadline, look for the first available slot (moving backward).
    - If a slot is available, schedule the job there and update the timeline.
4. **Calculate Total Profit:**
    - Add the profits of all scheduled jobs.

Example

Input:

- Jobs: (Deadline, Profit)
  [(2, 100), (1, 50), (2, 10), (1, 20), (3, 70)]

Output:

- Total Profit: 170
- Scheduled Jobs: [50, 100, 70]

Explanation:

- Job with profit 100 is scheduled at time 2.
- Job with profit 50 is scheduled at time 1.
- Job with profit 70 is scheduled at time 3.

Time Complexity

- Sorting jobs by profit: ***O(nlogn)***
- Assigning jobs to slots: ***O(n*d),*** where ***d*** is the maximum deadline.

Overall complexity: ***O(nlogn+n*d).***
If ***n*d≤n,*** the complexity simplifies to ***O(n^2).***

This greedy approach ensures that the solution is efficient and works well for problems where jobs need to be scheduled within specific deadlines for maximum profit.

Java Code:

```
import java.util.Arrays;
import java.util.Comparator;
```

```
class Job {
    int id;      // Job ID
    int deadline; // Deadline of the job
    int profit;   // Profit if the job is done before the deadline

    // Constructor
    public Job(int id, int deadline, int profit) {
        this.id = id;
        this.deadline = deadline;
        this.profit = profit;
    }
}

public class JobSequencing {

    // Function to find the maximum profit and job sequence
    public static void jobSequencing(Job[] jobs, int n) {
        // Step 1: Sort jobs by profit in descending order
        Arrays.sort(jobs, new Comparator<Job>() {
            @Override
            public int compare(Job j1, Job j2) {
                return j2.profit - j1.profit; // Sort by profit (descending)
            }
        });

        // Find the maximum deadline
        int maxDeadline = 0;
        for (Job job : jobs) {
            maxDeadline = Math.max(maxDeadline, job.deadline);
        }

        // Create a timeline array to track job scheduling
        int[] slots = new int[maxDeadline];
        Arrays.fill(slots, -1); // Initialize all slots as free (-1)
        int totalProfit = 0;

        // Step 2: Iterate over sorted jobs and assign them to slots
        for (Job job : jobs) {
            // Start from the job's deadline and move backward to find a free slot
            for (int i = job.deadline - 1; i >= 0; i--) {
                if (slots[i] == -1) { // If the slot is free
                    slots[i] = job.id; // Assign the job ID to the slot
                    totalProfit += job.profit; // Add the profit to total
                    break;
                }
            }
        }

        // Output the results
        System.out.println("Scheduled Jobs:");
        for (int i = 0; i < maxDeadline; i++) {
            if (slots[i] != -1) {
                System.out.print("Job " + slots[i] + " ");
            }
        }
        System.out.println("\nTotal Profit: " + totalProfit);
    }

    public static void main(String[] args) {
        // Example Jobs: {Job ID, Deadline, Profit}
        Job[] jobs = {
            new Job(1, 2, 100),
            new Job(2, 1, 50),
            new Job(3, 2, 10),
            new Job(4, 1, 20),
            new Job(5, 3, 70)
        };

        int n = jobs.length;

        // Call the job sequencing function
        jobSequencing(jobs, n);
    }
}
```

Explanation of Code

1. **Job Class**:
    - Represents a job with id, deadline, and profit.
2. **Sorting**:
    - Jobs are sorted in descending order of profit using Arrays.sort() with a custom comparator.
3. **Timeline Array**:
    - An array slots[] is used to represent available time slots.
    - Initially, all slots are marked as -1 (free).
4. **Job Assignment**:
    - For each job, starting from its deadline, we look for the nearest free slot.
    - If a slot is found, the job is scheduled, and its profit is added to the total profit.
5. **Output**:
    - Scheduled jobs and total profit are printed.

Example Output

Input Jobs:

```
[(1, 2, 100), (2, 1, 50), (3, 2, 10), (4, 1, 20), (5, 3, 70)]
```

Output:

```
Scheduled Jobs: Job 2 Job 1 Job 5

Total Profit: 220
```