5.4 MAXIMUM PROFIT FROM NON-OVERLAPPING JOBS

Question:

We have n jobs, where every job is scheduled to be done from startTime[i] to endTime[i], obtaining a profit of profit[i]. You're given the startTime, endTime and profit arrays, return the maximum profit you can take such that there are no two jobs in the subset with overlapping time range. If you choose a job that ends at time X you will be able to start another job that starts at time X.

AIM

To implement a greedy algorithm that selects the most profitable combination of non-overlapping jobs.

ALGORITHM

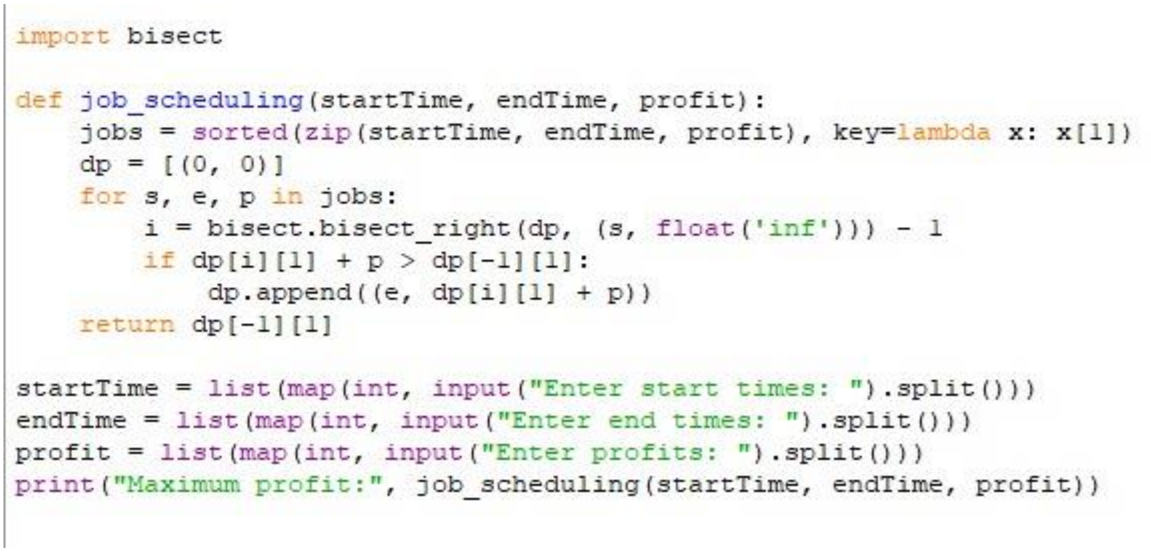
 Combine all jobs into a list of tuples: (start, end, profit)

1. Sort jobs by end time (earliest finishing jobs first)
2. Initialize an empty list selected_jobs to track chosen jobs
3. Iterate through sorted jobs:

- If the job's start time is ≥ end time of the last selected job, add it to the list
- Track total profit

This greedy method prioritizes earliest finishing jobs, which helps leave room for more jobs later.

PROGRAM

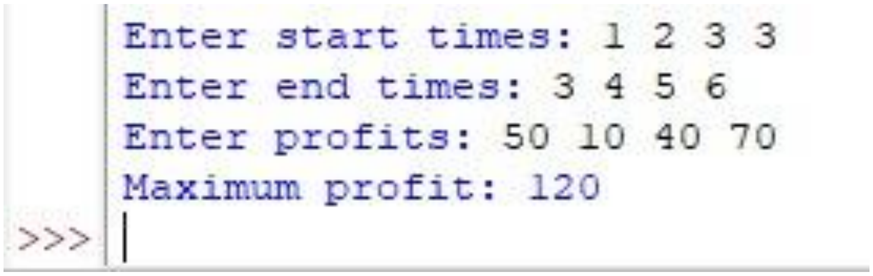


```
import bisect

def job_scheduling(startTime, endTime, profit):
    jobs = sorted(zip(startTime, endTime, profit), key=lambda x: x[1])
    dp = [(0, 0)]
    for s, e, p in jobs:
        i = bisect.bisect_right(dp, (s, float('inf'))) - 1
        if dp[i][1] + p > dp[-1][1]:
            dp.append((e, dp[i][1] + p))
    return dp[-1][1]

startTime = list(map(int, input("Enter start times: ").split()))
endTime = list(map(int, input("Enter end times: ").split()))
profit = list(map(int, input("Enter profits: ").split()))
print("Maximum profit:", job_scheduling(startTime, endTime, profit))
```

Input:

```
Enter start times: 1 2 3 3
Enter end times: 3 4 5 5
Enter profits: 50 10 40 70
```

Output:



```
Enter start times: 1 2 3 3
Enter end times: 3 4 5 6
Enter profits: 50 10 40 70
Maximum profit: 120
>>> |
```

RESULT:

Thus the program is successfully executed, and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity: O(n log n), due to sorting
- Space Complexity: O(1)