4.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.

Example 1:

Input: startTime = [1,2,3,3], endTime = [3,4,5,6], profit = [50,10,40,70]

Output: 120

Example 1:

Input: jobs = [3,2,3], k = 3

Output: 3

Program:

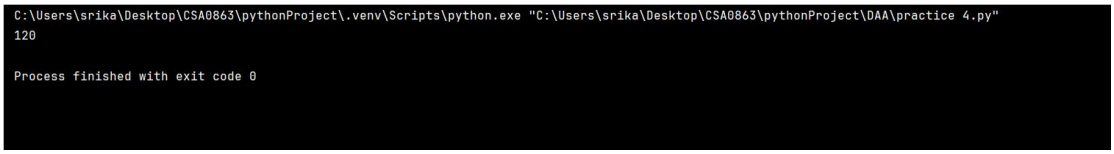
```
from bisect import bisect_right

def jobScheduling(startTime, endTime, profit):
    jobs = sorted(zip(startTime, endTime, profit), key=lambda x: x[1])
    n = len(jobs)
    dp = [0] * (n + 1)
    end_times = [job[1] for job in jobs]
    for i in range(1, n + 1):
        curr_start, curr_end, curr_profit = jobs[i - 1]
        idx = bisect_right(end_times, curr_start) - 1
        if idx != -1:
            dp[i] = max(dp[i - 1], dp[idx + 1] + curr_profit)
        else:
            dp[i] = max(dp[i - 1], curr_profit)
    return dp[n]
startTime = [1, 2, 3, 3]
endTime = [3, 4, 5, 6]
```

profit = [50, 10, 40, 70]

print(jobScheduling(startTime, endTime, profit))

Output:




Time complexity:

O(nlogn)