AMRITA VISHWA VIDYAPEETHAM

Second Year B.Tech

(Computer Science and Engineering)

Design and Analysis of Algorithms

TASK – 6

Name: Shaik Kolimi Dada Hussain

College Name: Amrita vishwa Vidyapeetham

Roll No : ch.sc.u4cse24144

Department: CSE-B

Academic Year : 2024-2028

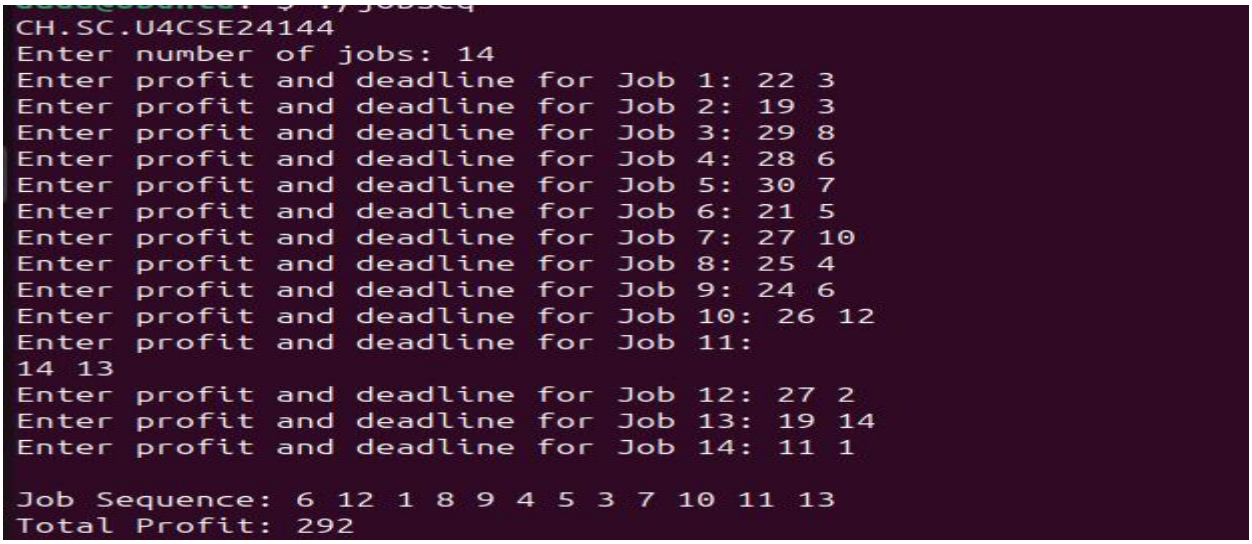
CODE:

```
// CH.SC.U4CSE24144

#include <stdio.h>
#include <stdlib.h>

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

void jobScheduling(Job jobs[], int n) {

    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (jobs[j].profit < jobs[j + 1].profit) {
                Job temp = jobs[j];
                jobs[j] = jobs[j + 1];
                jobs[j + 1] = temp;
            }
        }
    }

    int maxDeadline = 0;
    for (int i = 0; i < n; i++) {
        if (jobs[i].deadline > maxDeadline) {
            maxDeadline = jobs[i].deadline;
        }
    }
```

```
    int *result = (int *)malloc(maxDeadline * sizeof(int));
    int *slot = (int *)malloc(maxDeadline * sizeof(int));
    int totalProfit = 0;
    for (int i = 0; i < maxDeadline; i++) {
        slot[i] = 0;
    }
    for (int i = 0; i < n; i++) {
        for (int j = jobs[i].deadline - 1; j >= 0; j--) {
            if (slot[j] == 0) {
                result[j] = i;
                slot[j] = 1;
                totalProfit += jobs[i].profit;
                break;
            }
        }
    }
    printf("\nJob Sequence: ");
    for (int i = 0; i < maxDeadline; i++) {
        if (slot[i]) {
            printf("%d ", jobs[result[i]].id);
        }
    }
    printf("\nTotal Profit: %d\n", totalProfit);
    free(result);
    free(slot);
}
int main() {
    printf("CH.SC.U4CSE24144\n");
    int n;
    printf("Enter number of jobs: ");
    scanf("%d", &n);
```

```
    Job jobs[n];
    for (int i = 0; i < n; i++) {
        jobs[i].id = i + 1;
        printf("Enter profit and deadline for Job %d: ", jobs[i].id);
        scanf("%d %d", &jobs[i].profit, &jobs[i].deadline);
    }
    jobScheduling(jobs, n);
    return 0;
}
```

OUTPUT:



TIME AND SPACE COMPLEXITY:

The jobs are sorted using **bubble sort**, which uses two nested loops and performs about $n(n-1)/2$comparisons, giving a time complexity of **O(n²)**.
Finding the maximum deadline takes a single loop, which is **O(n)**.
The job scheduling part uses a nested loop where each of the $n$jobs may check up to Dslots (maximum deadline), giving **O(nD)**.
Since **O(n²)** dominates the other terms, the overall time complexity is **O(n²)**.

SPACE:

The program uses an array jobs of size n, which requires **O(n)** space.
Additionally, two dynamically allocated arrays slot and result of size D(maximum deadline) are used, requiring **O(D)** space.
Hence, the total auxiliary space complexity is **O(n + D)**