Design Analysis and Algorithm – Lab Work

Week 7

Question 1:Write a Program to perform Job Scheduling/Sequencing:

Code:

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

#define MAX 100
struct Job {
    int id;
    int deadline;
    int profit;
};

int compare(const void* a, const void* b) {
    struct Job* jobA = (struct Job*)a;
    struct Job* jobB = (struct Job*)b;
    return jobB->profit - jobA->profit;
}

int main() {
    int n;

    printf("Enter number of jobs: ");
    scanf("%d", &n);

    struct Job jobs[MAX];

    for(int i = 0; i < n; i++) {
        printf("\nJob %d\n", i + 1);
        printf("Enter Deadline: ");
        scanf("%d", &jobs[i].deadline);
        printf("Enter Profit: ");
        scanf("%d", &jobs[i].profit);
        jobs[i].id = i + 1;
    }

    qsort(jobs, n, sizeof(struct Job), compare);

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

```
    int slot[MAX] = {0};
    int totalProfit = 0;

    printf("\nSelected Jobs:\n");

    for(int i = 0; i < n; i++) {
        for(int j = jobs[i].deadline - 1; j >= 0; j--) {
            if(slot[j] == 0) {
                slot[j] = jobs[i].id;
                totalProfit += jobs[i].profit;
                printf("Job %d (Profit: %d)\n", jobs[i].id, jobs[i].profit);
                break;
            }
        }
    }

    printf("\nTotal Profit = %d\n", totalProfit);

    return 0;
}
```

Output:

```
PS E:\OneDrive - Amrita Vishwa Vidyapeetham- Chennai Campus\4th Semester\Design Analysis And Algorith
m\Week 7> gcc jobsequencing.c
PS E:\OneDrive - Amrita Vishwa Vidyapeetham- Chennai Campus\4th Semester\Design Analysis And Algorith
m\Week 7> ./a
Enter number of jobs: 14

Job 1
Enter Deadline: 3
Enter Profit: 22

Job 2
Enter Deadline: 3
Enter Profit: 19

Job 3
Enter Deadline: 8
Enter Profit: 29

Job 4
Enter Deadline: 6
Enter Profit: 28

Job 5
Enter Deadline: 7
Enter Profit: 30

Job 6
Enter Deadline: 5
Enter Profit: 21

Job 7
Enter Deadline: 10
Enter Profit: 27

Job 8
Enter Deadline: 4
Enter Profit: 25

Job 9
Enter Deadline: 6
Enter Profit: 24
```

```
Job 10
Enter Deadline: 12
Enter Profit: 26

Job 11
Enter Deadline: 13
Enter Profit: 14

Job 12
Enter Deadline: 2
Enter Profit: 27

Job 13
Enter Deadline: 14
Enter Profit: 19

Job 14
Enter Deadline: 1
Enter Profit: 11

Selected Jobs:
Job 5 (Profit: 30)
Job 3 (Profit: 29)
Job 4 (Profit: 28)
Job 7 (Profit: 27)
Job 12 (Profit: 27)
Job 10 (Profit: 26)
Job 8 (Profit: 25)
Job 9 (Profit: 24)
Job 1 (Profit: 22)
Job 6 (Profit: 21)
Job 13 (Profit: 19)
Job 11 (Profit: 14)

Total Profit = 292
```

Space Complexity (Worst Case):

The **space complexity** of the program is **O(n)**. This is because we use an array to store the **n jobs** and another array to store the available time slots up to the maximum deadline (which in the worst case can also be n). No additional dynamic memory allocation or recursion is used. Therefore, the total auxiliary space required grows linearly with the number of jobs, resulting in **O(n)** space complexity.

Time Complexity (Worst Case):

The **time complexity** of the given Job Sequencing with Deadlines program is mainly determined by three steps. First, the jobs are sorted in descending order of profit using qsort(), which takes **O(n log n)** time. Next, finding the maximum deadline requires a single traversal of the jobs, which takes **O(n)** time. The dominant part is the nested loop used for assigning jobs to slots. In the worst case, for each of the **n** jobs, we may check up to **n** slots before finding a free one, resulting in $\mathbf{O(n^2)}$ time. Since $\mathbf{O(n^2)}$ dominates the other terms, the overall time complexity of the program is $\mathbf{O(n^2)}$.