



**SCHOOL OF
COMPUTING**

Devadharshan.S

CH.SC.U4CSE24113

Week – 6

Date - 12/02/2026

Design and Analysis of Algorithm(23CSE211)

1. Job Scheduling

Code:

```
// CH.SC.U4CSE24113
#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.U4CSE24113\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:

```

devadharshan@devadharshan-HP-Pavilion-Laptop-15-eg3xxx: /media/devadharshan/New Volume/DAA/Lab/Week-6$ gcc Job_Scheduling.c -o Job_Scheduling
devadharshan@devadharshan-HP-Pavilion-Laptop-15-eg3xxx: /media/devadharshan/New Volume/DAA/Lab/Week-6$ ./Job_Scheduling
CH.SC.U4CSE24113
Enter number of jobs: 14
Enter profit and deadline for Job 1: 22 3
Enter profit and deadline for Job 2: 19 3
Enter profit and deadline for Job 3: 29 8
Enter profit and deadline for Job 4: 28 6
Enter profit and deadline for Job 5: 30 7
Enter profit and deadline for Job 6: 21 5
Enter profit and deadline for Job 7: 27 10
Enter profit and deadline for Job 8: 25 4
Enter profit and deadline for Job 9: 24 6
Enter profit and deadline for Job 10: 26 12
Enter profit and deadline for Job 11: 14 13
Enter profit and deadline for Job 12: 27 2
Enter profit and deadline for Job 13: 19 14
Enter profit and deadline for Job 14: 11 1

Job Sequence: 6 12 1 8 9 4 5 3 7 10 11 13
Total Profit: 292

```

Time Complexity: $O(n^2)$

Justification: The algorithm first uses a nested Bubble Sort to sort jobs based on profit, which takes $O(n^2)$. Following this, there is another nested loop structure where, for each of the n jobs, the algorithm scans through the slot array (up to the maximum deadline d) to find an available time slot. If the maximum deadline d is proportional to n , the total time complexity remains $O(n^2)$.

Space Complexity: $O(n+d)$

Justification: The algorithm allocates two dynamic arrays, result and slot, both of size maxDeadline (d). Additionally, it stores an array of n Job structures. Since it uses an iterative approach rather than recursion, there is no function call stack overhead. The total auxiliary space is determined by the number of jobs and the range of the deadlines.