Experiment No. 7
Job sequencing with deadline
Name : Vaidehi D. Gadag
Branch/Div.: Comps-1 (C47)
Date of Performance: 14/03/2024
Date of Submission: 28/03/2024

Experiment No. 7

Title: Job Sequencing with deadline

Aim:
To study and implement Job Sequencing with deadline Algorithm
Objective: To introduce Greedy based algorithms

Theory:

- Job sequencing algorithm is applied to schedule the jobs on a single processor to maximize the profits.
- The greedy approach of the job scheduling algorithm states that, "Given 'n' number of jobs with a starting time and ending time, they need to be scheduled in such a way that maximum profit is received within the maximum deadline".
- We are given n-jobs, where each job is associated with a deadline Di and a profit Pi if the job if finished before the deadline.
- We have single CPU with Non-Primitive Scheduling.
- With each job we assume arrival time is 0, burst time of each job requirement is 1.
- Select a Subset of 'n' jobs, such that, the jobs in the subset can be completed within deadline and generate maximum profit.

Strategy to solve job sequencing with deadlines problem:

Step 1: Arrange the list based on descending order of profits. Read the profits array from left to right.

Step 2: Fill up the job array using the deadlines.

Step 2.1: If the job array has vacant position at the location indicated by the deadline, then insert the pi at corresponding index in job array.

Step 2.2: If it is not vacant then search for the less than current deadline indexes

in the job array.

Step 2.3: If empty location is found the insert pi otherwise discard that job.

Step 3: Finally read the job array to get the optimal sequence.

Example:

Given the jobs, their deadlines and associated profits as shown-

Jobs	J1	J2	J3	J4	J5	J6
Deadlines	5	3	3	2	4	2
Profits	200	180	190	300	120	100

Answer the following questions-

1. Write the optimal schedule that gives maximum profit.
2. Are all the jobs completed in the optimal schedule?
3. What is the maximum earned profit?

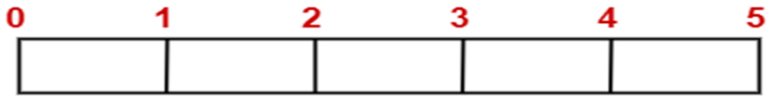
Solution:

Step-01: Sort all jobs in decreasing order of their profit.

Jobs	J4	J1	J3	J2	J5	J6
Deadlines	2	5	3	3	4	2
Profits	300	200	190	180	120	100

Step-02:

- Value of maximum deadline = 5.
- Draw a Gantt chart with maximum time on Gantt chart = 5 units



Gantt Chart

- Take each job one by one in the order they appear in Step-01 and place the job on Gantt chart as far as possible from 0.

Here, only job left is job J6 whose deadline is 2.

All the slots before deadline 2 are already occupied.

Thus, job J6 cannot be completed.

Maximum earned profit = Sum of profit of all the jobs in optimal schedule

= Profit of job J2 + Profit of job J4 + Profit of job J3

+ Profit of job J5 + Profit of job J1

= 180 + 300 + 190 + 120 + 200

= 990 units

Algorithm for Job sequencing with deadlines:

Step 1:
Sort all jobs in decreasing order of profit. — **O(n.logn)**

Step 2:
Iterate on jobs in decreasing order of profit.
For each job i =1 To N, do
Find a time slot i, such that
if (slot is empty and i < deadline and i is greatest)
Put the job in this slot and
Mark this slot filled.
Add the profit
else
if no such i exists, Then
Ignore the job.

Step 3:
Display the Profit

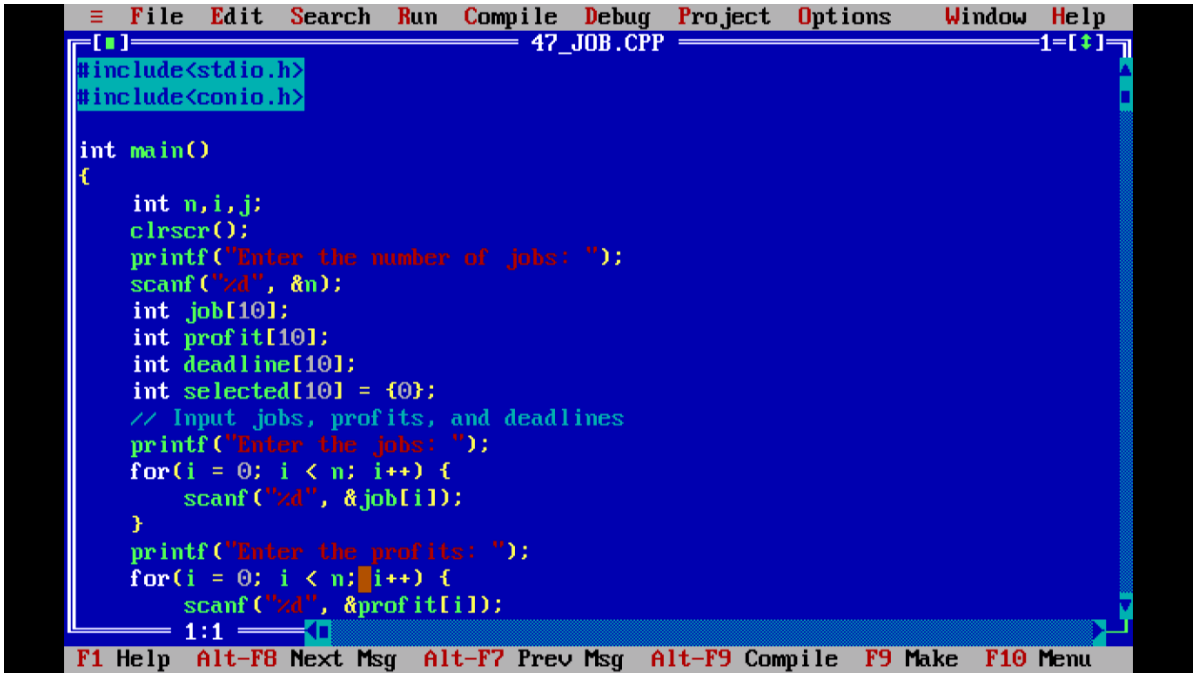
O(nxm)
When m=n
$O(n^2)$

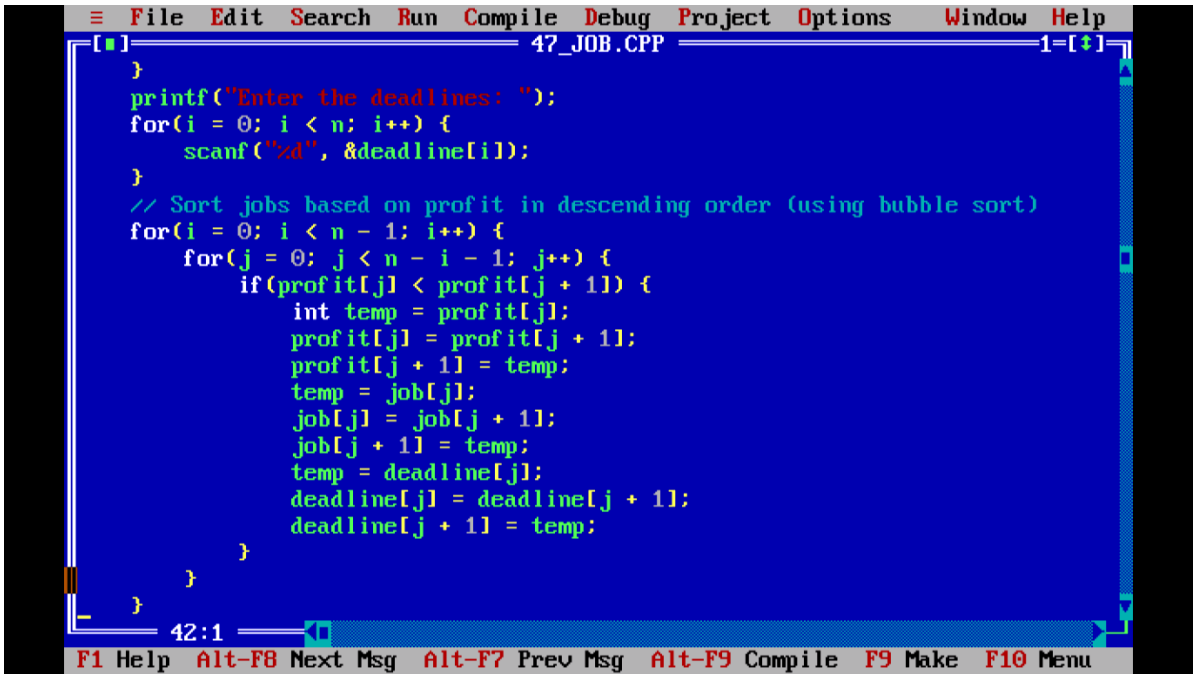
Time Complexity:
= $O(n.\log n) + O(n^2)$
= $\max(O(n.\log n), O(n^2)$
= $O(n^2)$

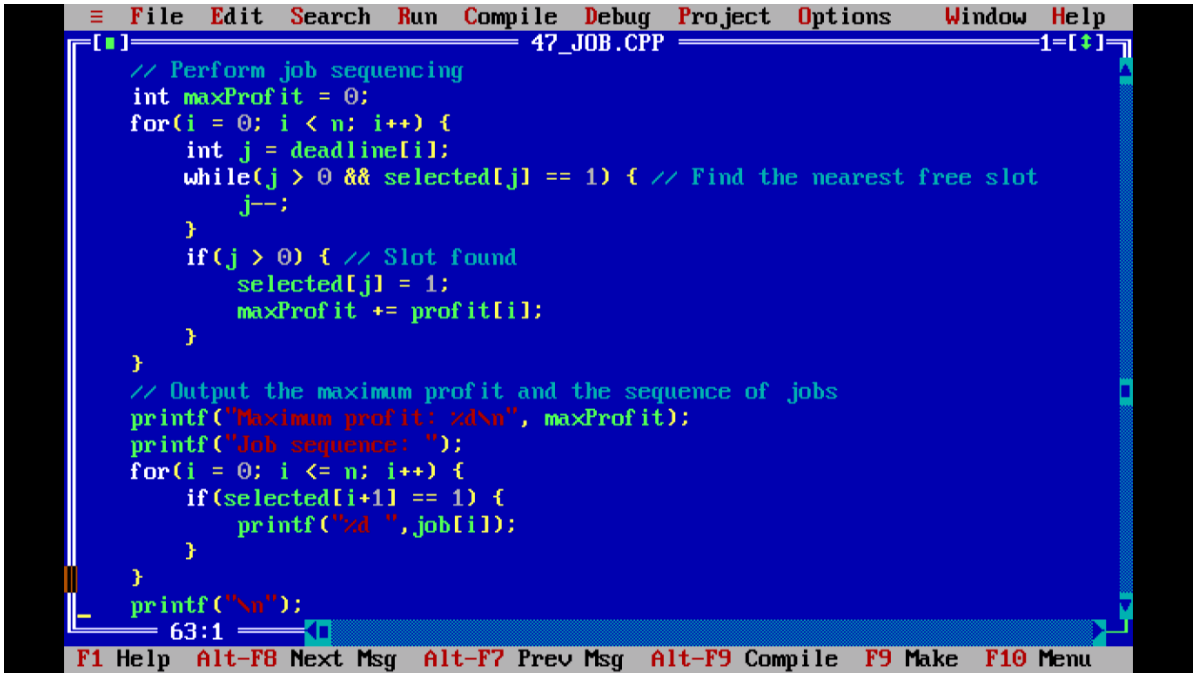
Code:

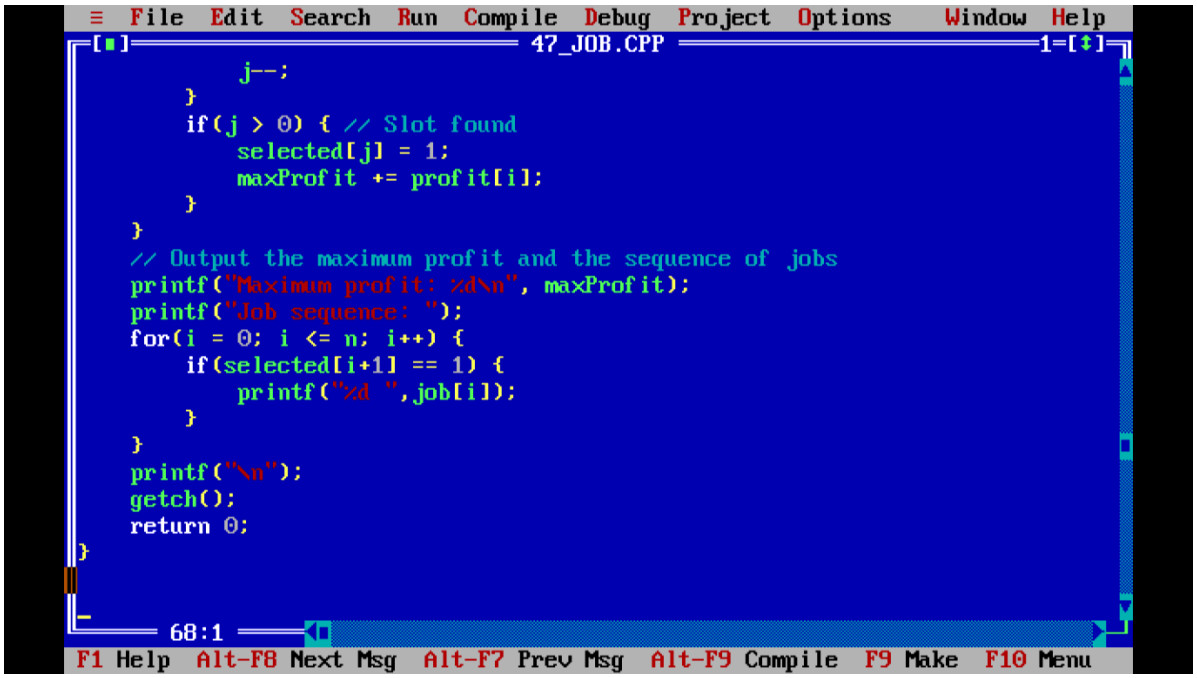
```
#include<stdio.h>
#include<conio.h>

int main()
{
   int n,i,j;
   clrscr();
   printf("Enter the number of jobs: ");
   scanf("%d", &n);
   int job[10];
   int profit[10];
   int deadline[10];
   int selected[10] = {0};
   // Input jobs, profits, and deadlines
   printf("Enter the jobs: ");
   for(i = 0; i < n; i++) {
        scanf("%d", &job[i]);
   }
   printf("Enter the profits: ");
   for(i = 0; i < n; i++) {
        scanf("%d", &profit[i]);
   }
   printf("Enter the deadlines: ");
   for(i = 0; i < n; i++) {
        scanf("%d", &deadline[i]);
   }
   // Sort jobs based on profit in descending order (using bubble sort)
   for(i = 0; i < n - 1; i++) {
        for(j = 0; j < n - i - 1; j++) {
           if(profit[j] < profit[j + 1]) {
                int temp = profit[j];
                profit[j] = profit[j + 1];
                profit[j + 1] = temp;
                temp = job[j];
                job[j] = job[j + 1];
```

```
                job[j + 1] = temp;
                temp = deadline[j];
                deadline[j] = deadline[j + 1];
                deadline[j + 1] = temp;
            }
        }
    }
    // Perform job sequencing
    int maxProfit = 0;
    for(i = 0; i < n; i++) {
        int j = deadline[i];
        while(j > 0 && selected[j] == 1) { // Find the nearest free slot
            j--;
        }
        if(j > 0) { // Slot found
            selected[j] = 1;
            maxProfit += profit[i];
        }
    }
    // Output the maximum profit and the sequence of jobs
    printf("Maximum profit: %d\n", maxProfit);
    printf("Job sequence: ");
    for(i = 0; i <= n; i++) {
        if(selected[i+1] == 1) {
            printf("%d ",job[i]);
        }
    }
    printf("\n");
    getch();
    return 0;
}
```


File Edit Search Run Compile Debug Project Options Window Help
47_JOB.CPP
#include<stdio.h>
#include<conio.h>
int main()
{
int n,i,j;
clrscr();
printf("Enter the number of jobs: ");
scanf("%d", &n);
int job[10];
int profit[10];
int deadline[10];
int selected[10] = {0};
// Input jobs, profits, and deadlines
printf("Enter the jobs: ");
for(i = 0; i < n; i++) {
scanf("%d", &job[i]);
}
printf("Enter the profits: ");
for(i = 0; i < n; i++) {
scanf("%d", &profit[i]);
1:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

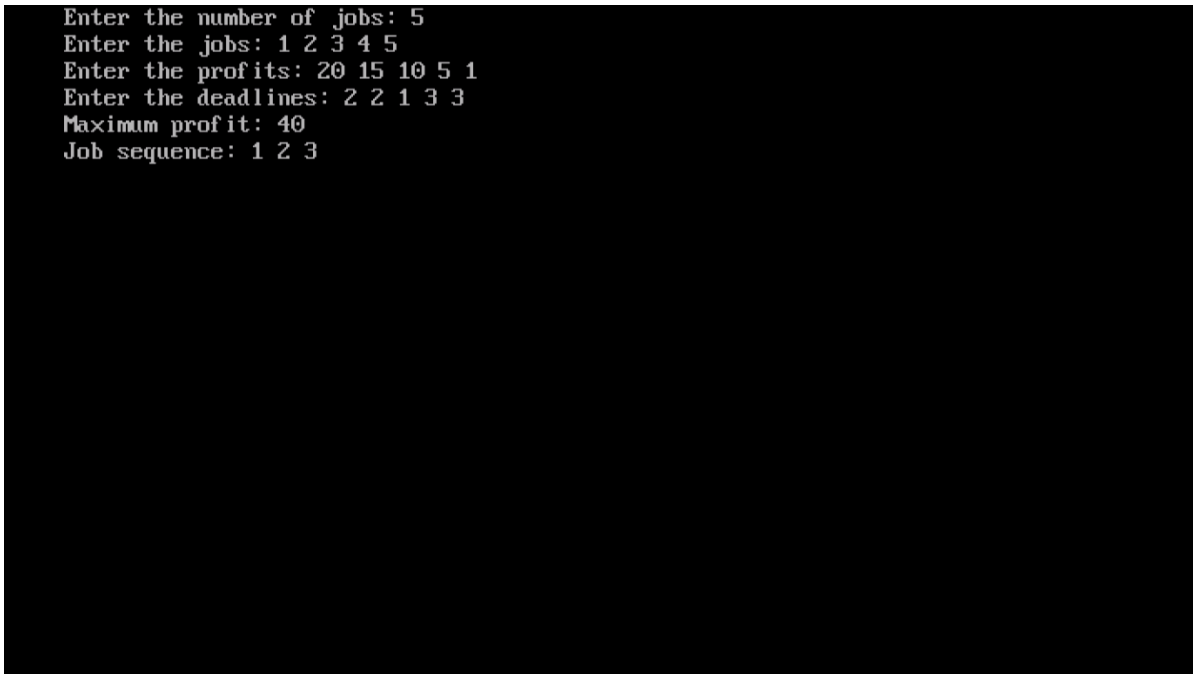

File Edit Search Run Compile Debug Project Options Window Help
47_JOB.CPP
}
printf("Enter the deadlines: ");
for(i = 0; i < n; i++) {
scanf("%d", &deadline[i]);
}
// Sort jobs based on profit in descending order (using bubble sort)
for(i = 0; i < n - 1; i++) {
for(j = 0; j < n - i - 1; j++) {
if(profit[j] < profit[j + 1]) {
int temp = profit[j];
profit[j] = profit[j + 1];
profit[j + 1] = temp;
temp = job[j];
job[j] = job[j + 1];
job[j + 1] = temp;
temp = deadline[j];
deadline[j] = deadline[j + 1];
deadline[j + 1] = temp;
}
}
}
42:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
47_JOB.CPP
// Perform job sequencing
int maxProfit = 0;
for(i = 0; i < n; i++) {
int j = deadline[i];
while(j > 0 && selected[j] == 1) { // Find the nearest free slot
j--;
}
if(j > 0) { // Slot found
selected[j] = 1;
maxProfit += profit[i];
}
}
// Output the maximum profit and the sequence of jobs
printf("Maximum profit: %d\n", maxProfit);
printf("Job sequence: ");
for(i = 0; i <= n; i++) {
if(selected[i+1] == 1) {
printf("%d ",job[i]);
}
}
printf("\n");
63:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
47_JOB.CPP
j--;
}
if(j > 0) { // Slot found
selected[j] = 1;
maxProfit += profit[i];
}
}
// Output the maximum profit and the sequence of jobs
printf("Maximum profit: %d\n", maxProfit);
printf("Job sequence: ");
for(i = 0; i <= n; i++) {
if(selected[i+1] == 1) {
printf("%d ",job[i]);
}
}
printf("\n");
getch();
return 0;
}
68:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

Output:

Enter the number of job: 5
Enter the jobs: 1 2 3 4 5
Enter the profits: 20 15 10 5 1
Enter the deadlines: 2 2 1 3 3
Maximum profit: 40
Job sequence: 1 2 3



Conclusion:

In conclusion, the job sequencing problem with a deadline is a classic example of a greedy approach problem, where the goal is to maximize the profit from a set of jobs by selecting a subset of jobs that can be completed within their deadlines. The solution's time complexity is O(N^2), where N is the number of jobs, as we need to sort the jobs based on their profit and then fill up the job array using the deadlines. The greedy approach ensures that the job with the maximum profit is selected first, and the overall solution is efficient and easy to implement.