AOA EXPERIMENT 6

Name: Mohammed Arhaan Ali
Div: S11
Roll No: 02

Aim: To study and implement Job Sequencing with deadlines using the Greedy method.

Theory:
Job scheduling algorithm is applied to schedule the jobs on a single processor to maximise 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”.

Greedy approach for job sequencing problem:

Greedily choose the jobs with maximum profit first, by sorting the jobs in decreasing order of their profit. This would help to maximise the total profit as choosing the job with maximum profit for every time slot will eventually maximise the total profit
Follow the given steps to solve the problem:

- Sort all jobs in decreasing order of profit.
- Iterate on jobs in decreasing order of profit.For each job , do the following :
    - Find a time slot i, such that slot is empty and i < deadline and i is greatest.Put the job in
      this slot and mark this slot filled.
    - If no such i exists, then ignore the job.

Job Scheduling Algorithm

Set of jobs with deadlines and profits are taken as an input with the job scheduling algorithm and scheduled subset of jobs with maximum profit are obtained as the final output.

Algorithm

Step1 − Find the maximum deadline value from the input set of jobs.
Step2 − Once, the deadline is decided, arrange the jobs in descending order of their profits.
Step3 − Selects the jobs with highest profits, their time periods not exceeding the maximum deadline.
Step4 − The selected set of jobs are the output.

Examples

Consider the following tasks with their deadlines and profits. Schedule the tasks in such a way that they produce maximum profit after being executed −

S. No.	1	2	3	4	5
Jobs	J1	J2	J3	J4	J5
Deadlines	2	2	1	3	4
Profits	20	60	40	100	80

Step 1
Find the maximum deadline value, dm, from the deadlines given.
dm = 4.

Step 2
Arrange the jobs in descending order of their profits.

S. No.	1	2	3	4	5
Jobs	J4	J5	J2	J3	J1
Deadlines	3	4	2	1	2
Profits	100	80	60	40	20

The maximum deadline, d_m, is 4. Therefore, all the tasks must end before 4.
Choose the job with the highest profit, J4. It takes up 3 parts of the maximum deadline.
Therefore, the next job must have the time period 1.
Total Profit = 100.
Step 3
The next job with the highest profit is J5. But the time taken by J5 is 4, which exceeds the deadline by 3. Therefore, it cannot be added to the output set.
Step 4
The next job with the highest profit is J2. The time taken by J5 is 2, which also exceeds the deadline by 1. Therefore, it cannot be added to the output set.
Step 5
The next job with higher profit is J3. The time taken by J3 is 1, which does not exceed the given deadline. Therefore, J3 is added to the output set.
Total Profit: 100 + 40 = 140

Step 6
Since the maximum deadline is met, the algorithm comes to an end. The output set of jobs scheduled within the deadline are {J4, J3} with the maximum profit of 140.

Program:

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

// A structure to represent a job
typedef struct Job {

    char id; // Job Id
    int dead; // Deadline of job
    int profit; // Profit if job is over before or on
                // deadline
```

```
} Job;

// This function is used for sorting all jobs according to
// profit
int compare(const void* a, const void* b)
{
    Job* temp1 = (Job*)a;
    Job* temp2 = (Job*)b;
    return (temp2->profit - temp1->profit);
}

// Find minimum between two numbers.
int min(int num1, int num2)
{
    return (num1 > num2) ? num2 : num1;
}

// Returns maximum profit from jobs
void printJobScheduling(Job arr[], int n)
{
    // Sort all jobs according to decreasing order of profit
    qsort(arr, n, sizeof(Job), compare);

    int result[n]; // To store result (Sequence of jobs)
    bool slot[n]; // To keep track of free time slots

    // Initialize all slots to be free
    for (int i = 0; i < n; i++)
        slot[i] = false;

    // Iterate through all given jobs
    for (int i = 0; i < n; i++) {

        // Find a free slot for this job (Note that we start
        // from the last possible slot)
        for (int j = min(n, arr[i].dead) - 1; j >= 0; j--) {
```

```
            // Free slot found
            if (slot[j] == false) {
                result[j] = i; // Add this job to result
                slot[j] = true; // Make this slot occupied
                break;
            }
        }
    }

    // Print the result
    for (int i = 0; i < n; i++)
        if (slot[i])
            printf("%c ", arr[result[i]].id);
}

// Driver's code
int main()
{
    Job arr[] = { { 'a', 2, 100 },
                  { 'b', 1, 19 },
                  { 'c', 2, 27 },
                  { 'd', 1, 25 },
                  { 'e', 3, 15 } };
    int n = sizeof(arr) / sizeof(arr[0]);

    printf(
        "Following is maximum profit sequence of jobs \n");

    // Function call
    printJobScheduling(arr, n);
    return 0;
}
```

Output:

```
PS C:\Users\arhaa\OneDrive\Desktop\AOA> cd "c:\Users\arhaa\OneDri
 } ; if ($?) { .\jobsequencing }
Following is maximum profit sequence of jobs
c a e
PS C:\Users\arhaa\OneDrive\Desktop\AOA>
```

Conclusion: Thus we have successfully implemented Job Sequencing with deadlines using the Greedy method.