**Experiment 5**

***Aim:*** Perform Job Sequencing with a deadline using the Greedy Approach using C/C++.

***5.1 Objective:***

In the job sequencing problem, the objective is to find a sequence of jobs, which is completed within their deadlines and gives maximum profit.

***5.2 Program Logic:***

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.

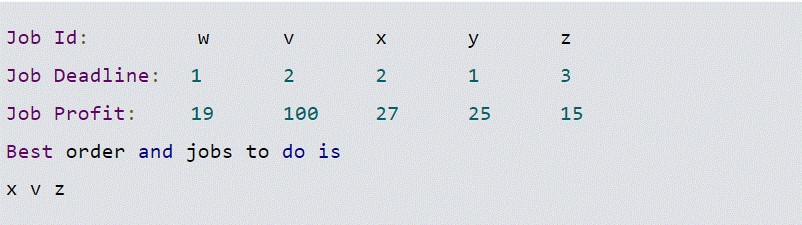
The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.

1. Begin
2. Sort all the jobs based on profit Pi so
3. P1 > P2 > P3 …………………………….>=Pn
4. d = maximum deadline of job in A
5. Create array S[1,…………………,d]
6. For i=1 to n do
7. Find the largest job x
8. For j=i to 1
9. If ((S[j] = 0) and (x deadline<= d))
10. Then
11. S[x] = i;
12. Break;
13. End if
14. End for
15. End for
16. End

5.3 ***Program Code:***

|  |
| --- |
| *#include <stdio.h>*  *#include <stdlib.h>*  *#include <stdbool.h>* ***//Job Structure*** *typedef struct { char id; int deadline; int profit;)}job;*  ***// function to compare two jobs based on their profit. returns true if job b's profit < job a's profit***    *int compareJob(const Job \*a, const Job \*b){ return b->profit - a->profit;*  *}*  ***//function finds the best job sequence*** *void bestJob(Job jobs[],int sizeOfJobs){*  ***//null char array***  *char jobsToDo[5]= {'\0'};*    *for(int i=0, k=0; i<sizeOfJobs; i++){*  *k = jobs[i].deadline-1;*  ***//Searching backwards the empty date nearest to deadline***  *while(jobsToDo[k] != '\0' && k >= 0){ k--;*  *}*  ***//if empty date found, set the job*** *if(k != -1)*  *jobsToDo[k]= jobs[i].id;*  *}*  ***//output the final job sequence*** *printf("\nBest order and jobs to do is: "); int idx=0; while(jobsToDo[idx] != '\0'){ printf("%c ",jobsToDo[idx]);*  *idx++;*  *}}*  ***//function to display the jobs table*** *void display(Job jobs[], int n){*  *printf("Job Id: \t"); for(int i=0; i<n; i++){ printf("%c \t",jobs[i].id);* |
| *}*  *printf("\n");*    *printf("Job Deadline: \t"); for(int i=0; i<n; i++){*  *printf("%d \t",jobs[i].deadline);*  *}*  *printf("\n");*    *printf("Job Profit: \t"); for(int i=0; i<n; i++){*  *printf("%d \t",jobs[i].profit);*  *}*  *printf("\n");*  *}*  *int main()*  *{*  ***//intialize the jobs***  *Job jobs[]= {{'w', 1, 19}, {'v', 2, 100}, {'x', 2, 27},*  *{'y', 1, 25}, {'z', 3, 15}};*  ***//display the jobs data***  *display(jobs,5);*    ***//sorting jobs[] w.r.t their profit***  *qsort(jobs,5,sizeof(jobs[0]),compareJob);*    *bestJob(jobs,5); return 0;*  *}* |

***5.4 Conclusion:***



***5.5 Analysis:***

The time complexity of this algorithm is O(n^2).