K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Department of Computer Engineering

Batch: B1 **Roll No.: 1711072**

Experiment No. 5

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of Job sequencing with deadline algorithm using Greedy strategy

Objective: To learn the Greedy strategy of solving the problems for different types of problems

CO to be achieved:

Sr. No	Objective
CO 1	Compare and demonstrate the efficiency of algorithms using asymptotic complexity notations.
CO 2	Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies.
CO 3	Analyze and solve problems for different string matching algorithms.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://lcm.csa.iisc.ernet.in/dsa/node184.htm**
4. **http://students.ceid.upatras.gr/~papagel/project/kruskal.htm**

5. **http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/kruskalAlgor.html**
6. **http://lcm.csa.iisc.ernet.in/dsa/node183.html**
7. **http://students.ceid.upatras.gr/~papagel/project/prim.htm**
8. **http://www.cse.ust.hk/~dekai/271/notes/L07/L07.pdf**

Pre Lab/ Prior Concepts:

Data structures, Concepts of algorithm analysis

Theory: The greedy method suggests that one can devise an algorithm that work in stages, considering one input at a time. At each stage, a decision is made regarding whether a particular input is in an optimal solution. This is done by considering the inputs in an order determined by some selection procedure. If the inclusion of the next input into the partially constructed optimal solution will result in an infeasible solution, then this input is not added to the partial solution. Otherwise, it is added. The selection procedure itself is based on some optimization measures may be plausible for a given problem. Most of these, however, will result in algorithms that generate suboptimal solutions. This version of the greedy technique is called the **subset paradigm**.

Control Abstraction:

```
SolType Greedy (Type s [ ], int n)

// a[1:n] contains the n inputs.

{       SolType solution = EMPTY;
        // Initialize the solution.
        For (int i=1; I<=n; i++) {
                Type x = Select (a) ;

        If Feasible (solution , x)

        Solution = Union (solution , x) ;

}

return solution ;

}
```

Problem Definition:

There are n jobs to be processed on a processor, Each job i has a deadline $d_i \geq 0$, and profit $p_i \geq 0$.Profit p_i is earned if and only if job is completed within the deadline.Only one processor is available and it can process only one job at a time.Job is completed if it is processed for a unit time, so the minimum possible deadline for any job is 1.So a schedule S should be find consisting of a sequence of job "slots", that maximizes profit.

A set of jobs is feasible if there exists at least one schedule that allows all the jobs to be executed within their deadlines. An optimal solution is a feasible solution with maximum profit value.

Algorithm:

```
algorithm JS (d, J, n)  {
        d[0] = J[0] = 0  // Initialize
      k   = J[1] = 1  // Job 1 is already chosen
      for i = 2 to n do  // Decreasing order of P jobs are scheduled
      {       r = k //checking feasibility of insertion for i at position
              while( d[ J[r]] > d[i]) and (d[J[r]]≠r)) do
                      r = r-1
              if ( d[ J[r]] <= d[i] and (d[i]>r)
              {
                for q = k to r+1 do //insert I into J[]
                        J[q+1] = J[q]
                J[r+1] = i
                k = k+1       }
      }
  return k
}
```

Code (in Python):

```
prof_dl=[[1,3,1],[2,5,3],[3,20,4],[4,18,3],[5,0,2],[6,6,1],[7,30,2]]
def sequencing(prof_dl):
  job_size=(max(prof_dl, key=lambda x: x[2]))[2]
  profit=[0]*job_size
  isEmpty,job_seq=[True]*job_size,['']*job_size
  prof_dl=sorted(prof_dl, key= lambda x: x[1],reverse=True)
  for i in range(len(prof_dl)):
    for j in range(min(job_size-1,prof_dl[i][2]-1),-1,-1):
      if isEmpty[j] is True:
        isEmpty[j],profit[j],job_seq[j]=False,prof_dl[i][1],prof_dl[i][0]
        break
  print('Jobs\tSlot\tProfit')
  for i in range(job_size):
    print( ' ',job_seq[i],'\t',i+1,'\t\t',profit[i])
  print('Profit is: ',sum(profit))
sequencing(prof_dl)
```

Example Problem:

Job Number	Profit	Deadline
J1	3	1
J2	5	3
J3	20	4
J4	18	3
J5	0	2
J6	6	1
J7	30	2

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Jobs    Slot    Profit
  6       1       6
  7       2       30
  4       3       18
  3       4       20
Profit is:  74
```

Analysis of Job sequencing with deadline algorithm:

The main algorithm is in the function sequencing(prof_dl). The main logic is two nested for loops. The outer loop runs for total number of jobs while the inner loop runs for maximum deadline. Hence, the time complexity for almost close values of number of jobs and maximum deadline will be:

$$\mathbf{T(n)=O(n^2)}$$

CONCLUSION:

The job sequencing algorithm ran successfully for various test cases and gave the optimized solution.