

|  |
| --- |
| **Title:** Implementation ofJob 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. **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. **https://www.mpi-inf.mpg.de/~mehlhorn/ftp/ShortestPathSeparator.pdf**
4. **en.wikipedia.org/wiki/Shortest\_path\_problem**
5. **www.cs.princeton.edu/~rs/AlgsDS07/15ShortestPaths.pdf**



**Pre Lab/ Prior Concepts:**

Data structures, Concepts of algorithm analysis

**Topic: GREEDY METHOD**

**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 di ≥0, and profit pi ≥0. Profit pi 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

}

**Example Problem:**

Find an optimal solution to the problem of job sequencing with deadline where n=4, (p1,p2,p3,p4)=(100,10,15,27)and (d1,d2,d3,d4)=(2,1,2,1)

|  |  |  |  |
| --- | --- | --- | --- |
| **Si No:** | **Feasible Solution** | **Processing Sequence** | **Profit value** |
| 1. | (1,2) | ( 2,1) | 110 |
| 2. | (1,3) | (1,3) or (3,1) | 115 |
| 3. | (1,4) | (4,1) | 127 is the optimal one |
| 4 | (2,3) | (2,3) | 25 |
| 5 | (3,4) | (4,3) | 42 |
| 6 | (1) | (1) | 100 |
| 7 | (2) | (2) | 10 |
| 8 | (3) | (3) | 15 |
| 9 | (4) | (4) | 27 |

**Solution:**

Program:  
import java.util.\*;

class jobs

{

int dl,no,pft,ct;

public static void main(String[] args)

{

Scanner s =new Scanner(System.in);

System.out.println("Enter the no of jobs");

int n=s.nextInt();

jobs a[]=new jobs[n];

for(int i=0;i<n;i++)

{

a[i]=new jobs();

a[i].no=i+1;

System.out.print("Job "+(i+1)+":\nEnter the deadline:");

a[i].dl=s.nextInt();

System.out.print("Enter the profit:");

a[i].pft=s.nextInt();

}

for(int i=0;i<n;i++)

{

for(int j=0;j<n-i-1;j++)

{

if(a[j].pft<a[j+1].pft)

{

jobs temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

int l=1,profit=0;

for(int i=0;i<n;i++)

{

if(a[i].dl>=l)

{

profit=profit+a[i].pft;

l++;

a[i].ct=-1;

}

}

System.out.print("The jobs ");

for(int i=0;i<5;i++)

{

if(a[i].ct==-1)

{

System.out.print(a[i].no+", ");

}

}

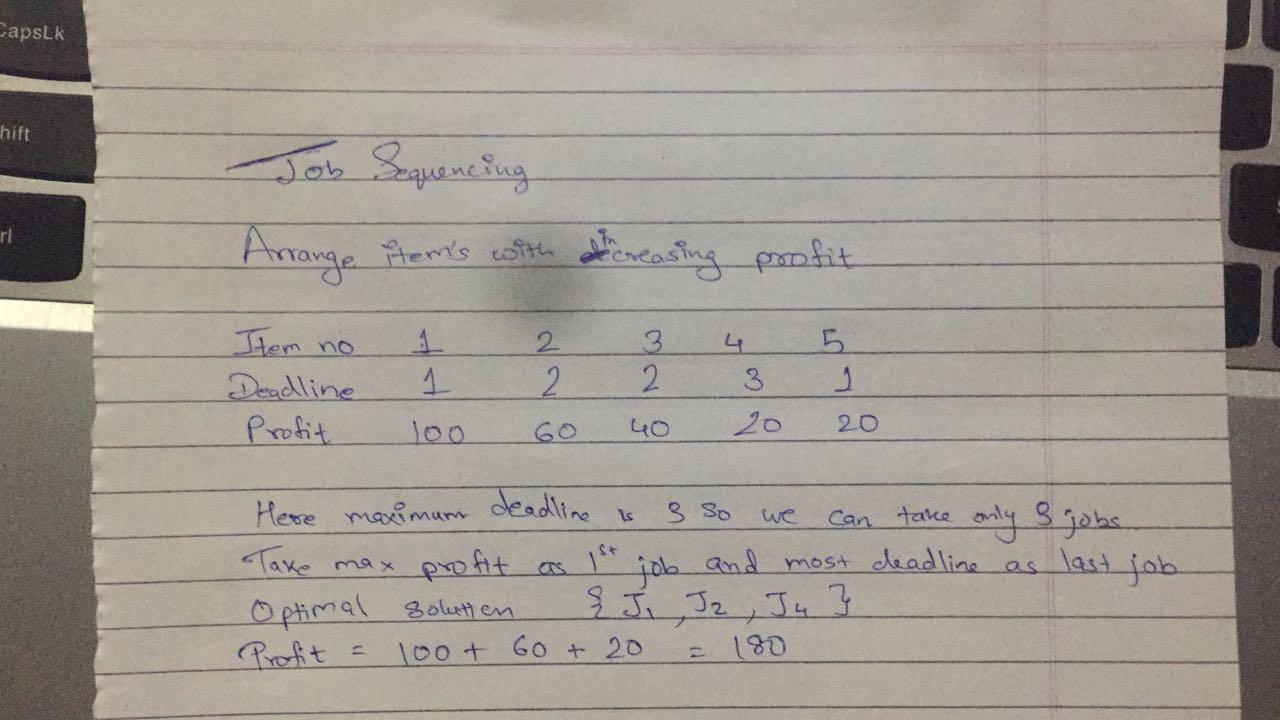
System.out.print(" is considered.\n");

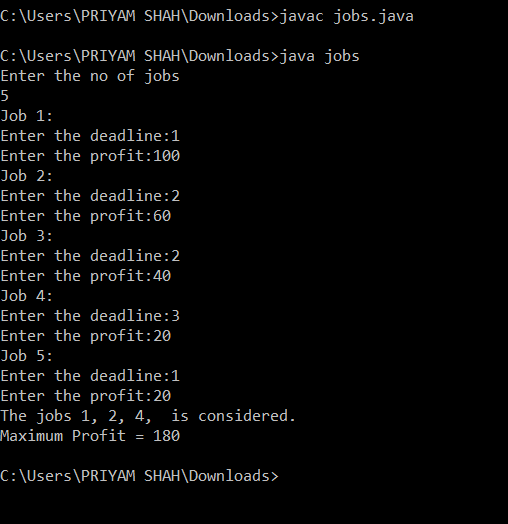
System.out.println("Maximum Profit = "+profit);

}

}

Output:





**Time Complexity for Job Sequencing with Deadline Problem**

**This would be an O(n2) algorithm**

**• By using disjoint set data structure, it becomes nearly O(n)**

**• We invoke n set finds and unions in our greedy algorithm**

**CONCLUSION:**

implementation job sequencing successfully using greedy strategy