Job Sequencing with deadlines using greedy method

```
import java.util.*;
public class job
{
  public static void main(String args[])
  {
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter the number of Jobs");
    int n=sc.nextInt();
    String a[]=new String[n];
    int b[]=new int[n];
    int c[]=new int[n];
    for(int i=0;i<n;i++)
    {
      System.out.println("Enter the Jobs");
      a[i]=sc.next();
      System.out.println("Enter the Profit");
      b[i]=sc.nextInt();
      System.out.println("Enter the DeadLine");
      c[i]=sc.nextInt();
    }
    System.out.println("--Arranged Order--");
    System.out.print("Jobs:    ");
    for(int i=0;i<n;i++)
```

```
{
  System.out.print(a[i]+" ");
}
System.out.println();
System.out.print("Profit:  ");
for(int i=0;i<n;i++)
{
  System.out.print(b[i]+" ");
}
System.out.println();
System.out.print("DeadLine:");
for(int i=0;i<n;i++)
{
  System.out.print(c[i]+" ");
}
for(int i=0;i<n-1;i++)
{
  for(int j=i+1;j<n;j++)
  {
      if(b[i]<b[j])
      {
          int temp=b[i];
          b[i]=b[j];
           b[j]=temp;

          temp=c[i];
          c[i]=c[j];
           c[j]=temp;

          String temp1=a[i];
          a[i]=a[j];
           a[j]=temp1;
      }
  }
}
System.out.println();
System.out.println("--Sorted Order--");
System.out.print("Jobs:    ");
for(int i=0;i<n;i++)
```

```
{
  System.out.print(a[i]+" ");
}
System.out.println();
System.out.print("Profit:  ");
for(int i=0;i<n;i++)
{
  System.out.print(b[i]+" ");
}
System.out.println();
System.out.print("DeadLine:");
for(int i=0;i<n;i++)
{
  System.out.print(c[i]+" ");
}
System.out.println();
int max=c[0];
for(int i=0;i<n;i++)
{
  if(c[i]>max)
  {
    max=c[i];
  }
}
String x[]=new String[max];
int xx[]=new int[max];
int profit=0;
for(int i=0;i<n;i++)
{
  int pp=c[i];
  pp=pp-1;
  if(x[pp]==null )
  {
    x[pp]=a[i];
    profit+=b[i];
  }
  else
  {
    while(pp!=-1)
```

```
        {
          if(x[pp]==null)
          {
            x[pp]=a[i];
            profit+=b[i];
            break;
          }
            pp=pp-1;
        }
      }
    }
    for(int i=0;i<max;i++)
    {
        System.out.print("-->"+x[i]);
    }
    System.out.println();
    System.out.print("Profit Earned"+profit);
 }
}
```

```
// Dijkstra's Algorithm in Java


public class Dijkstra {


  public static void dijkstra(int[][] graph, int source) {
    int count = graph.length;
    boolean[] visitedVertex = new boolean[count];
    int[] distance = new int[count];
    for (int i = 0; i < count; i++) {
      visitedVertex[i] = false;
      distance[i] = Integer.MAX_VALUE;
    }
```

```
  // Distance of self loop is zero
  distance[source] = 0;
  for (int i = 0; i < count; i++) {

    // Update the distance between neighbouring vertex and source vertex
    int u = findMinDistance(distance, visitedVertex);
    visitedVertex[u] = true;

    // Update all the neighbouring vertex distances
    for (int v = 0; v < count; v++) {
      if (!visitedVertex[v] && graph[u][v] != 0 && (distance[u] + graph[u][v] < distance[v])) {
        distance[v] = distance[u] + graph[u][v];
      }
    }
  }
  for (int i = 0; i < distance.length; i++) {
    System.out.println(String.format("Distance from %s to %s is %s", source, i, distance[i]));
  }

}

// Finding the minimum distance
private static int findMinDistance(int[] distance, boolean[] visitedVertex) {
  int minDistance = Integer.MAX_VALUE;
```

```
    int minDistanceVertex = -1;
    for (int i = 0; i < distance.length; i++) {
      if (!visitedVertex[i] && distance[i] < minDistance) {
        minDistance = distance[i];
        minDistanceVertex = i;
      }
    }
    return minDistanceVertex;
  }

  public static void main(String[] args) {
    int graph[][] = new int[][] { { 0, 0, 1, 2, 0, 0, 0 }, { 0, 0, 2, 0, 0, 3, 0 }, { 1, 2, 0, 1, 3, 0, 0 },
        { 2, 0, 1, 0, 0, 0, 1 }, { 0, 0, 3, 0, 0, 2, 0 }, { 0, 3, 0, 0, 2, 0, 1 }, { 0, 0, 0, 1, 0, 1, 0 } };
    Dijkstra T = new Dijkstra();
    T.dijkstra(graph, 0);
  }
}
```