LAB ASSIGNMENT – 3

DESIGN ANALYSIS AND ALGORITHMS

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

1Q) Implementation of travelling sales person problem using brute force approach

CODE :

```
package Lab3a;
import java.util.*;
public class TravellingSalesManBruteForce {
	static int V = 4;
	static int travellingSalesmanProblem(int graph[][], int s)
	{
	   ArrayList<Integer> vertex = new ArrayList<Integer>();
	   for (int i = 0; i < V; i++)
	       if (i != s)
	           vertex.add(i);
	   int min_path = Integer.MAX_VALUE;
	   do
	   {
	       int current_pathweight = 0;
	       int k = s;
	       for (int i = 0; i < vertex.size(); i++)
	       {
	           current_pathweight += graph[k][vertex.get(i)];
	           k = vertex.get(i);
	       }
	       current_pathweight += graph[k][s];
	       min_path = Math.min(min_path, current_pathweight);
	   } while (findNextPermutation(vertex));
	   return min_path;
	}
	public static ArrayList<Integer> swap(ArrayList<Integer> data,int left, int
right){
	   int temp = data.get(left);
	   data.set(left, data.get(right));
	   data.set(right, temp);
	   return data;
	}
	public static ArrayList<Integer> reverse(ArrayList<Integer> data, int left,
int right)
	{
	   while (left < right)
```

```
    {
        int temp = data.get(left);
        data.set(left++, data.get(right));
        data.set(right--, temp);
    }
    return data;
}
public static boolean findNextPermutation(ArrayList<Integer> data)
{
    if (data.size() <= 1)
        return false;
    int last = data.size() - 2;
    while (last >= 0)
    {
        if (data.get(last) < data.get(last + 1))
        {
            break;
        }
        last--;
    }
    if (last < 0)
        return false;
    int nextGreater = data.size() - 1;
    for (int i = data.size() - 1; i > last; i--)
    {
        if (data.get(i) > data.get(last))
        {
            nextGreater = i;
            break;
        }
    }
    data = swap(data,nextGreater, last);
    data = reverse(data, last + 1, data.size() - 1);
    return true;
}
public static void main(String args[])
{
    int graph[][] = {{0, 12, 18, 20},
                     {8, 0, 45, 37},
                     {15, 36, 0, 40},
                     {22, 25, 38, 0}};
    int s = 0;
    System.out.println(travellingSalesmanProblem(graph, s));
}
}
```

```java
package Lab3a;
import java.util.*;
public class TravellingSalesManBruteForce {
    static int V = 4;
    static int travellingSalesmanProblem(int graph[][], int s)
    {
       ArrayList<Integer> vertex = new ArrayList<Integer>();
       for (int i = 0; i < V; i++)
           if (i != s)
               vertex.add(i);
       int min_path = Integer.MAX_VALUE;
       do
       {
           int current_pathweight = 0;
           int k = s;
           for (int i = 0; i < vertex.size(); i++)
           {
               current_pathweight += graph[k][vertex.get(i)];
               k = vertex.get(i);
           }
           current_pathweight += graph[k][s];
           min_path = Math.min(min_path, current_pathweight);
       } while (findNextPermutation(vertex));
       return min_path;
    }
    public static ArrayList<Integer> swap(ArrayList<Integer> data,int left, int right){
       int temp = data.get(left);
       data.set(left, data.get(right));
       data.set(right, temp);
       return data;
    }
    public static ArrayList<Integer> reverse(ArrayList<Integer> data, int left, int right)
    {
       while (left < right)
       {
           int temp = data.get(left);
           data.set(left++, data.get(right));
           data.set(right--, temp);
       }
       return data;
    }
    public static boolean findNextPermutation(ArrayList<Integer> data)
    {
```

```
43      {
44          if (data.size() <= 1)
45              return false;
46          int last = data.size() - 2;
47          while (last >= 0)
48          {
49              if (data.get(last) < data.get(last + 1))
50              {
51                  break;
52              }
53              last--;
54          }
55          if (last < 0)
56              return false;
57          int nextGreater = data.size() - 1;
58          for (int i = data.size() - 1; i > last; i--)
59          {
60              if (data.get(i) > data.get(last))
61              {
62                  nextGreater = i;
63                  break;
64              }
65          }
66          data = swap(data,nextGreater, last);
67          data = reverse(data, last + 1, data.size() - 1);
68          return true;
69      }
70      public static void main(String args[])
71      {
72          int graph[][] = {{0, 12, 18, 20},
73                           {8, 0, 45, 37},
74                           {15, 36, 0, 40},
75                           {22, 25, 38, 0}};
76          int s = 0;
77          System.out.println(travellingSalesmanProblem(graph, s));
78      }
79      }
```

OUTPUT :

```
<terminated> TravellingSalesManBruteForce [Java Applicati
91
```

2Q) implementation of knapsack problem using brute force approach

CODE :

```
package Lab3b;
public  class KnapsackBruteForce{

          static int max(int a, int b)
          {

             return (a > b) ? a : b;
          }

          static int knapSack(int W, int wt[], int val[], int n)
          {

              if (n == 0 || W == 0)

                   return 0;

              if (wt[n - 1] > W)
                  return knapSack(W, wt, val, n - 1);

              else
                  return max(val[n - 1]
                             + knapSack(W - wt[n - 1], wt,
                                        val, n - 1),
                             knapSack(W, wt, val, n - 1));
          }

          public static void main(String args[])
          {

             int val[] = new int[] { 40, 150, 180 };
              int wt[] = new int[] { 30, 40, 60 };

              int W = 150;
              int n = val.length;

              System.out.println(knapSack(W, wt, val, n));
          }
      }
```

```java
package Lab3b;
public  class KnapsackBruteForce{

        static int max(int a, int b)
        {

            return (a > b) ? a : b;
        }

        static int knapSack(int W, int wt[], int val[], int n)
        {

            if (n == 0 || W == 0)

                return 0;

            if (wt[n - 1] > W)
                return knapSack(W, wt, val, n - 1);

            else
                return max(val[n - 1]
                           + knapSack(W - wt[n - 1], wt,
                                      val, n - 1),
                           knapSack(W, wt, val, n - 1));
        }

        public static void main(String args[])
        {

            int val[] = new int[] { 40, 150, 180 };
            int wt[] = new int[] { 30, 40, 60 };

            int W = 150;
            int n = val.length;

            System.out.println(knapSack(W, wt, val, n));
        }
    }
```

OUTPUT :

```
<terminated> KnapsackBruteForce (1) [Java Application]
370
```