CSE- 3004 LAB- Assignment

Academic year: 2020-2021 **Semester:** WIN

Faculty Name: Dr. D Sumathi Mam **Date:** 10 /11/2021

Student name: M.Taran **Reg. no.:** 19BCE7346

Optimal BST using dynamic programming

TSP AND COMPARISON

1) Parents entered the school for joining their daughter in 2nd grade. The class teacher in that grade has some excellent capabilities to make the students to think out of the box. She follows the concept of arranging the children in her class in a way.

she arranges in such a way that there will be 5 in a row. Boys and girls will occupy certain positions. Find all the possible ways of arranging the boys and girls.

Constraint 1 : Girl cannot be seated in the middle.

Implement backtracking for this problem.

Code :

```
public class Main{
public static String gender(int val) {
if(val == 1) {
return "B";
}
return "G";
}
public static void main(String args[]) {
System.out.println("Possible seating arrangement");
for(int i = 0;i < 2;i++) {
for(int j = 0;j < 2;j++) {
for(int k = 0;k < 2;k++) {
for(int l = 0;l < 2;l++) {
for(int m = 0;m < 2;m++) {
if((i==0||j==0||l==0||m==0)&&(k==1)) {
System.out.println(gender(i)+" "+gender(j)+" "+gender(k)+" "+gender(l)+"
```

```
"+gender(m));
}
}
}
}
}
}
}
}
```

Output :

```
Possible seating arrangement
G G B G G
G G B G B
G G B B G
G G B B B
G B B G G
G B B G B
G B B B G
G B B B B
B G B G G
B G B G B
B G B B G
B G B B B
B B B G G
B B B G B
B B B B G


...Program finished with exit code 0
Press ENTER to exit console.
```

2)

Kennedy has a plan to visit all cities. He can visit the city only once. Distance is given for all cities. Find the shortest possible route that he visits each city exactly once and returns to the origin city.

Solve this problem using backtracking and dynamic approach. Compare both the results. Find which solution will be a better one.

Code :

Tsp using Backtracking :

```
class Main
{
```

```
static int tsp(int[][] graph, boolean[] v,
int currPos, int n,
int count, int cost, int ans)
{
if (count == n && graph[currPos][0] > 0)
{
ans = Math.min(ans, cost + graph[currPos][0]); return ans;

{
if (v[i] == false && graph[currPos][i] > 0)
{
ans = tsp(graph, v, i, n, count + 1,
cost + graph[currPos][i], ans);
}
}
return ans;
}

// Driver code
public static void main(String[] args)
{
int[][] graph = {{0, 4, 12, 7},

{5, 0, 0, 18},
{11, 0, 0, 6},
{10, 2, 3, 0}};

// Boolean array to check if a node

int ans = Integer.MAX_VALUE;
}
}
```

Output :

```
25


...Program finished with exit code 0
Press ENTER to exit console.
```

Tsp using Dynamic Programming :

```java
import java.util.*;
class Main{
     static int V = 4;
 static int travllingSalesmanProblem(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, 4, 12, 7},
                   {5, 0, 0, 18},
                   {11, 0, 0, 6},
                   {10, 2, 3, 0}};
  int s = 0;
  System.out.println(
  travllingSalesmanProblem(graph, s));
}
}
```

Output :

```
15


...Program finished with exit code 0
Press ENTER to exit console.
```