习题10

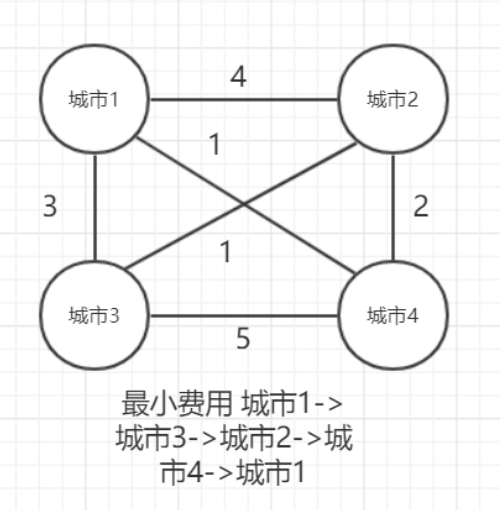
* 1. 旅行商问题

一个售货员必须访问n个城市，恰好访问每个城市一次，并最终回到出发城市。  
 售货员从城市i到城市j的旅行费用是一个整数，旅行所需的全部费用是他旅行经过的的各边费用之和，而售货员希望使整个旅行费用最低。

1.1.1.算法描述

（等价于求图的最短哈密尔顿回路问题）令G=(V, E)是一个带权重的有向图，顶点集V=(v0, v1, ..., vn-1)。从图中任一顶点vi出发，经图中所有其他顶点一次且只有一次，最后回到同一顶点vi的最短路径。

1.1.2.抽象模型



1.1.3.算法描述

通过排列算法，进行暴力试，如果能够通过，那么加入数组。

1.1.4.算法实现（Java）

// 输出排列次序并求距离

public static void arrangementPath(City[] city, int[][] distance) {

tmpArr.add(0);

arrangement(city.length);

int min = Integer.MAX\_VALUE;

System.out.println("所有的组合序列：");

for (int i = 0; i < Arr.size(); i++) {

System.out.print(i + ":\t");

int dis = 0;

for (int j = 0; j < Arr.get(i).size(); j++) {

System.out.print(city[Arr.get(i).get(j)].getName() + "->");

if (j + 1 == Arr.get(i).size())

dis += getDistance(Arr.get(i).get(j), Arr.get(i).get(0), distance);

else

dis += getDistance(Arr.get(i).get(j), Arr.get(i).get(j + 1), distance);

}

if (dis < min) {

min = dis;

shortPath.clear();

shortPath.add(i);

}else if(dis==min) {

shortPath.add(i);

}

System.out.print(city[0].getName() + " " + dis + "\n");

}

System.out.println("最短的距离为: "+min);

System.out.println("最短的路径有: "+shortPath.toString());

}

// 对n个数组排列

public static void arrangement(int n) {

int[] list = new int[n - 1];

for (int i = 0; i < list.length; i++)

list[i] = i + 1;

arrangement(list.length, list);

//System.out.println(Arr.toString());

}

// 排列

public static void arrangement(int k, int[] arr) {

if (k == 1) {

for (int i = 0; i < arr.length; i++) {

tmpArr.add(arr[i]);

Arr.add(new ArrayList<Integer>(tmpArr));

tmpArr.remove((Object) arr[i]);

}

} else if (k > 1) {

for (int i = 0; i < arr.length; i++) { // 按顺序挑选一个元素

tmpArr.add(arr[i]); // 添加选到的元素

arrangement(k - 1, removeArrayElements(arr, tmpArr.toArray(new Integer[1]))); // 没有取过的元素，继续挑选

tmpArr.remove((Object) arr[i]);

}

} else {

return;

}

}