联系

- prim和djs都是基于贪心的。
- 都有一个distTo数组记录权值。

区别：

- prim是要遍历所有的点。
- djs求源点v到其它各点最短路径就行。如：求v->w的最短路径，这条路径不必遍历所有的点，v->w的权值和最小就行
- distTo数组的区别
  - prim的distTo是某点到最小生成树的距离。是到这棵树的距离，没有指定到那个点。
  - djs的distTo是某点到源点的最短距离。
- 总结
  - prim的distTo是记录点到树的距离
  - djs的distTo是记录点到源点的距离
  - prim的更新是更新以w为中间点到生成树的距离,距离更小就更新,否则不更新.
  - djs的更新是以w为中间点到源点v的距离,距离更小就更新,否则不更新.

为什么这个贪心的策略是正确的?

- 请查看课本第四章图论的内容。

自己写一遍，在回顾一下，差不多就懂了

附上简化版代码

```
public void primSimple(int v, double[][] matrix) {
    int len = matrix[v].length;
    double[] distTo = new double[len];
    boolean[] marked = new boolean[len];
    double minWeight = 0.0;
    LinkedQueue<Integer> queue = new LinkedQueue<Integer>();//队列
    marked[v] = true;
    queue.enqueue(v);
    for (int i = 0; i < len; i++) {
        distTo[i] = matrix[v][i];
    }
    while (queue.size() < len) {//还存在为加入的点
        double min = Double.MAX_VALUE;
        int index = 0;
        // 找距离生成树最近的点
        for (int i = 0; i < len; i++) {
            if (distTo[i] != 0 && min > distTo[i]) {
                index = i;
                min = distTo[i];
            }
        }
        minWeight += distTo[index];
        distTo[index] = 0;
        marked[index] = true;
```

```
            queue.enqueue(index);
            for (int i = 0; i < len; i++) {
                if (!marked[i] && distTo[i] > matrix[index][i]) {
                    distTo[i] = matrix[index][i];
                }
            }
        }
        System.out.println(minWeight);
    }

    ==========================================
    //精简版适用于刷题
    public void djs(int v,double[][]matrix) {
        int vertexNumber = matrix[v].length;//顶点数目
        double[]distTo = new double[vertexNumber];//v距离各点的权值
        double[]weights = new double[vertexNumber];//记录v-i的最终的最短权值
        //初始化distTo
        for(int i=0;i<vertexNumber;i++) {
            distTo[i] = matrix[v][i];
        }
        distTo[v] = 0; weights[v] = 0;//初始化顶点
        int count = 0;
        while(count < vertexNumber-1) {
            double min = Double.MAX_VALUE;
            int index = 0;
            //找最小的边 加入进行
            for(int i = 0; i < vertexNumber; i++) {
                if(distTo[i] != 0 && min > distTo[i]) {
                    min = distTo[i]; index = i;
                }
            }
            //更新权值
            for(int i = 0 ;i < vertexNumber; i++) {
                if(distTo[i] > distTo[index] + matrix[index][i]) {
                    distTo[i] = distTo[index] + matrix[index][i];
                }
            }
            weights[index] = distTo[index];
            distTo[index] = 0;
            count++;
        }
        System.out.println(Arrays.toString(weights));
    }
```

代码效率优化建议
查找最小值使用优先队列

```
PriorityQueue<Integer> priorityqueue = new PriorityQueue<Integer>();//默认是小顶堆
```