算法设计与分析实验报告

倪玮昊

2020211346

3.1 题目描述

使用 **Prim 算法**计算给定无向图上以顶点 1 为根的最小生成树边权和。

3.2 输入格式

输入文件名为 prim.in。
第一行包含两个正整数 V、E，分别代表顶点数与边数。
接下来 E 行包含三个正整数 u、v、w，代表 u、v 之间存在一条权值为 w 的**无向边**。

3.3 输出格式

输出文件名为 prim.out。输出共一行。
第一行包含一个整数代表最小生成树的权值和。

3.4 输入输出样例

prim.in	prim.out
4 5 1 2 2 1 3 2 1 4 3 2 3 4 3 4 3	7

3.5 数据范围

$0 < V \leq 5000$,
$0 < E \leq 2 \times 10^5$,
$0 < u, v \leq V$,
$0 < w \leq 2 \times 10^5$。

3.6 说明/提示

保证图中没有自环。

二、实验过程

算法思路

对图G(V,E)设置集合S，存放已访问的顶点，然后每次从集合V-S中选择与集合S的最短距离最小的一个顶点(记为u)，访问并加入集合S。之后，令顶点u为中介点，优化所有从u能到达的顶点v与集合S之间的最短距离。执行n次(n为顶点个数)，直到集合S已包含所有顶点。

关键函数及代码段的描述

找单独一个已经加入集合的点和非集合内的点的最短距离

```
int prim(int start,bool *M){
    int tem;
    int min=200001;
    for(int i=0;i<n;i++){
        if(M[i]==false){
            if(min>a[start][i]){
                min=a[start][i];
                tem=i;
            }
        }
    }
    return tem;
}
```

将集合里的点遍历,看看谁的与集合外的点的距离最短

```
    while(count<n){
        int min=200001;
        rep(i, 0, count-1){
            int l=prim(N[i],M);
            if(min>a[i][l]){
                min=a[i][l];
                tem=l;
            };

        }
            total=total+min;
            M[tem]=true;
            N[count]=tem;
            count++;

    }
```

算法时间及空间复杂性分析

*时间复杂度为O(N^2),*两级循环找最小点

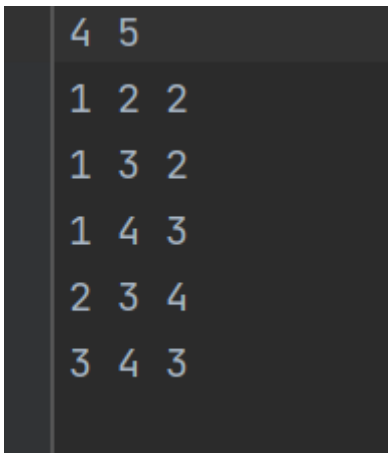
三、实验结果

程序执行环境及运行方式

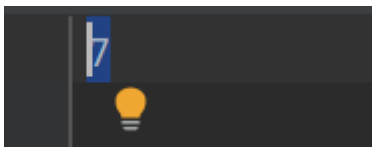
win11,CLion

程序执行示例

输入:



输出:



四、 实验总结

和迪杰斯特拉算法很像,区别就是迪杰斯特拉找的是某点和某点的最短距离,prim找某个点集合和点最短距离

五、 算法源代码

```
#include <deque>
#include <algorithm>
#include <iostream>
#include <fstream>
using namespace std;
int n;
int lineNum;
int a[5001][5001];
int tem;
#define rep(i, a, b) for (int i = a; i <= b; ++i)
void startA(){
    for(int i=0;i<5001;i++){
        for(int i1=0;i1<5001;i1++){
            a[i][i1]=200001;
        }
    }

}

int prim(int start,bool *M){
    int tem;
    int min=200001;
    for(int i=0;i<n;i++){
        if(M[i]==false){
            if(min>a[start][i]){
                min=a[start][i];
                tem=i;
            }
        }
```

```
    }
    return tem;
}
int main()
{
    fstream fin("prim.in", ios::in), fout("prim.out", ios::out);
    fin >> n;
    tem=200000*n+1;
    fin>>lineNum;
    startA();
    int start,end;
    rep(i, 0, lineNum-1) {
        fin >> start;
        fin >> end;
        fin >> a[start-1][end-1];
        a[end-1][start-1]=a[start-1][end-1];
    }
    int N[n];
    N[0]=0;
    bool M[n];
    rep(i, 0, n-1){
        M[i]=false;
    }
    M[0]=true;
    int count=1;
    int total=0;
    int tem;
    while(count<n){
        int min=200001;
        rep(i, 0, count-1){
            int l=prim(N[i],M);
            if(min>a[i][l]){
                min=a[i][l];
                tem=l;
            };

        }
            total=total+min;
            M[tem]=true;
            N[count]=tem;
            count++;

    }
    fout<<total;


    return 0;

}
```