Hard Performance The2nd-shortestPath Student

Date： Nov 27th

1. Introduction

Hard The 2nd-shortest Path：

Find the length of the second shortest path from 1 to M, and print the path through points. The number of sides is N.1<=M<=1000,1<=N<=5000

2. Algorithm Specification

Obviously, what the title requires is to find the length of the second short path and the node number, so the dijsktra algorithm can be used to calculate the shortest and second short circuits from the first point to the i-th point. Shorter than the shortest path.

Pseudo Code:

```
for(k=1;k<=n*2;k++){
    mi=INF;x=0;
    for(i=1;i<=n;i++)
        for(j=0;j<=1;j++)
            if(!b[i][j]&&mi>f[i][j]){
                mi=f[i][j];
                x=i+j*n;
            }//Find the target point of the shortest path that
```

has not been marked

if(x==0)break;//End the loop if the target point is not found

b[fi(x)][se(x)]=1;//Mark the target point

for(i=1;i<=n;i++)

if(f[fi(x)][se(x)]+a[fi(x)][i]<f[i][0]){

f[i][1]=f[i][0];p[i][1]=p[i][0];

f[i][0]=f[fi(x)][se(x)]+a[fi(x)][i];p[i][0]=x;

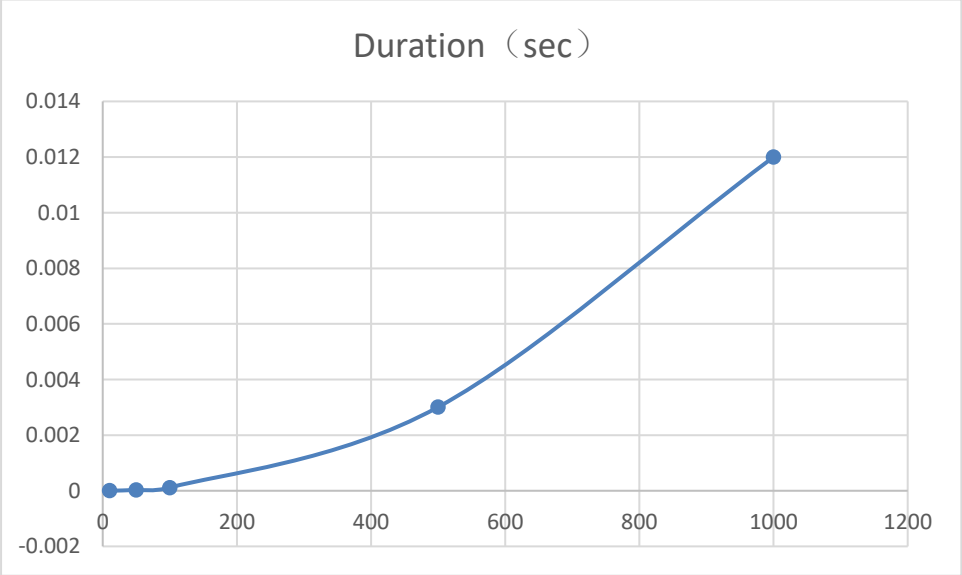
}else if(f[fi(x)][se(x)]+a[fi(x)][i]<f[i][1]){

f[i][1]=f[fi(x)][se(x)]+a[fi(x)][i];p[i][1]=x;

}//Use the current shortest path target point as a relay point to update the path

}

3. Testing Results



M	10	50	100	500	1000
Iterations（K）	10000	1000	1000	100	1
Ticks	13	31	115	301	12
Total Time（sec）	0.013	0.031	0.115	0.301	0.012
Duration（sec）	0.0000013	0.000031	0.000115	0.00301	0.012

4. Analysis and Comments

Obviously, I used the dijkstra algorithm, and the algorithm is not nested in other loops, so the time complexity of the program is O(M^2+N), and the space complexity is O(M^2)

5. Appendix: Source Code

```
#include<stdio.h>
#define fi(x) ((x-1)%n+1)
#define se(x) ((x-1)/n)
#define N 1005
const int INF=(1<<29)-1;//max
int n,m,a[N][N],f[N][2],p[N][2],b[N][2],x,y,z,mi,i,j,k;
//a[i][j] represents the path length of i->j, if a[i][j]==INF, it means no connection
//f[i][0] represents the shortest short-circuit length of 1->i,
//f[i][1] represents the secondary short-circuit length of 1->i
//p[i][0] represents the shortest path of 1->i to the previous node of point i,
//p[i][1] represents the secondary short circuit of 1->i to the previous node of point i
//b[i][0] indicates whether the shortest path of 1->i is used as a
```

```
relay path,
//b[i][1] indicates whether the shortest path of 1->i is used as a
relay path
void dfs(int x){
    if(fi(x)!=1)dfs(p[fi(x)][se(x)]);
    printf(" %d",fi(x));
}
int main(){
    scanf("%d%d",&n,&m);//Read n and m
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            a[i][j]=INF;//Assign the maximum distance between i
and j to indicate no path
    while(m--){
        scanf("%d%d%d",&x,&y,&z);//Read path
        a[x][y]=a[y][x]=z;//Write path to a array
    }
    for(i=1;i<=n;i++)f[i][0]=a[1][i],p[i][0]=1,f[i][1]=INF;
    //Initialize the shortest distance and the second shortest
distance from 1 to i
    f[1][0]=0;b[1][0]=1;//Shortest path of pretreatment 1->1
    for(k=1;k<=n*2;k++){
```

```
        mi=INF;x=0;
        for(i=1;i<=n;i++)
            for(j=0;j<=1;j++)
                if(!b[i][j]&&mi>f[i][j]){
                    mi=f[i][j];
                    x=i+j*n;
                }//Find the target point of the shortest path that has not been marked
        if(x==0)break;//End the loop if the target point is not found
        b[fi(x)][se(x)]=1;//Mark the target point
        for(i=1;i<=n;i++)
            if(f[fi(x)][se(x)]+a[fi(x)][i]<f[i][0]){
                f[i][1]=f[i][0];p[i][1]=p[i][0];
                f[i][0]=f[fi(x)][se(x)]+a[fi(x)][i];p[i][0]=x;
            }else if(f[fi(x)][se(x)]+a[fi(x)][i]<f[i][1]){
                f[i][1]=f[fi(x)][se(x)]+a[fi(x)][i];p[i][1]=x;
            }//Use the current shortest path target point as a relay point to update the path
    }
    printf("%d",f[n][1]);//Output secondary short circuit length
    dfs(p[n][1]);printf(" %d\n",n);//Output secondary
```

```
short-circuit path
    return 0;
}
```