PM9:00

2021.12.8

导游图

目录
CONTENTS

GraphicalDesign

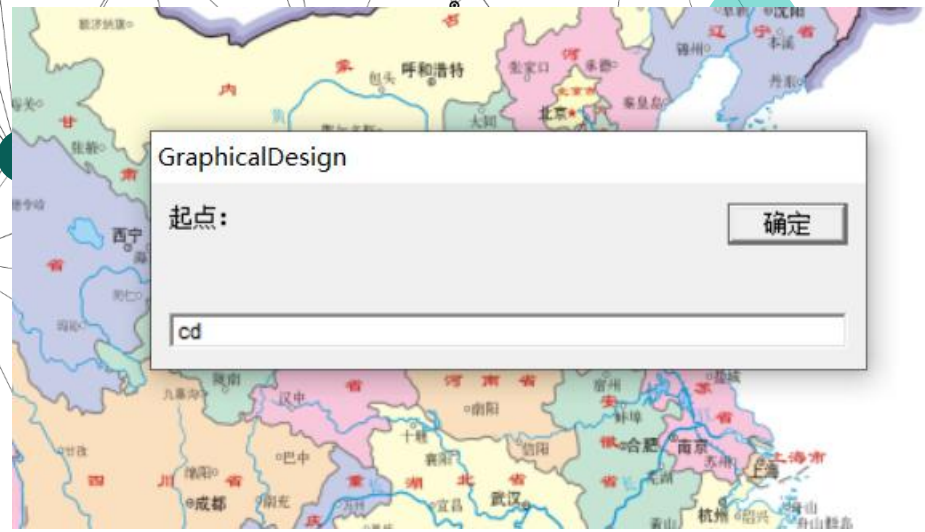
导游图

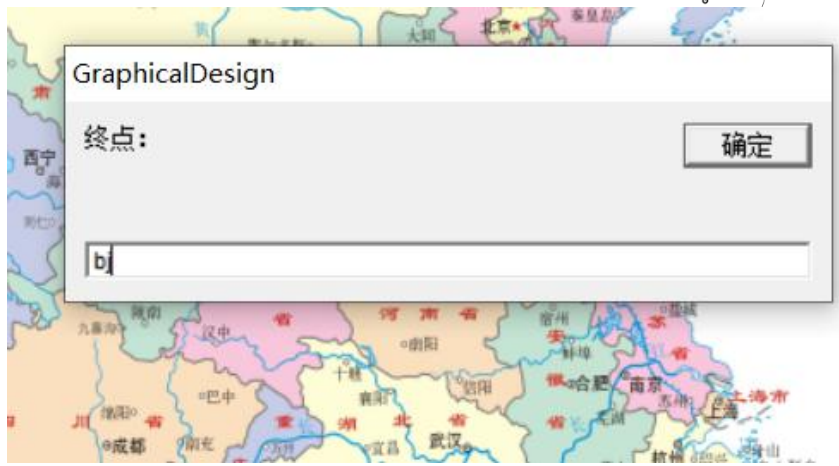
开始导游

城市介绍

退出程序

开始导游


GraphicalDesign
起点：
确定
cd

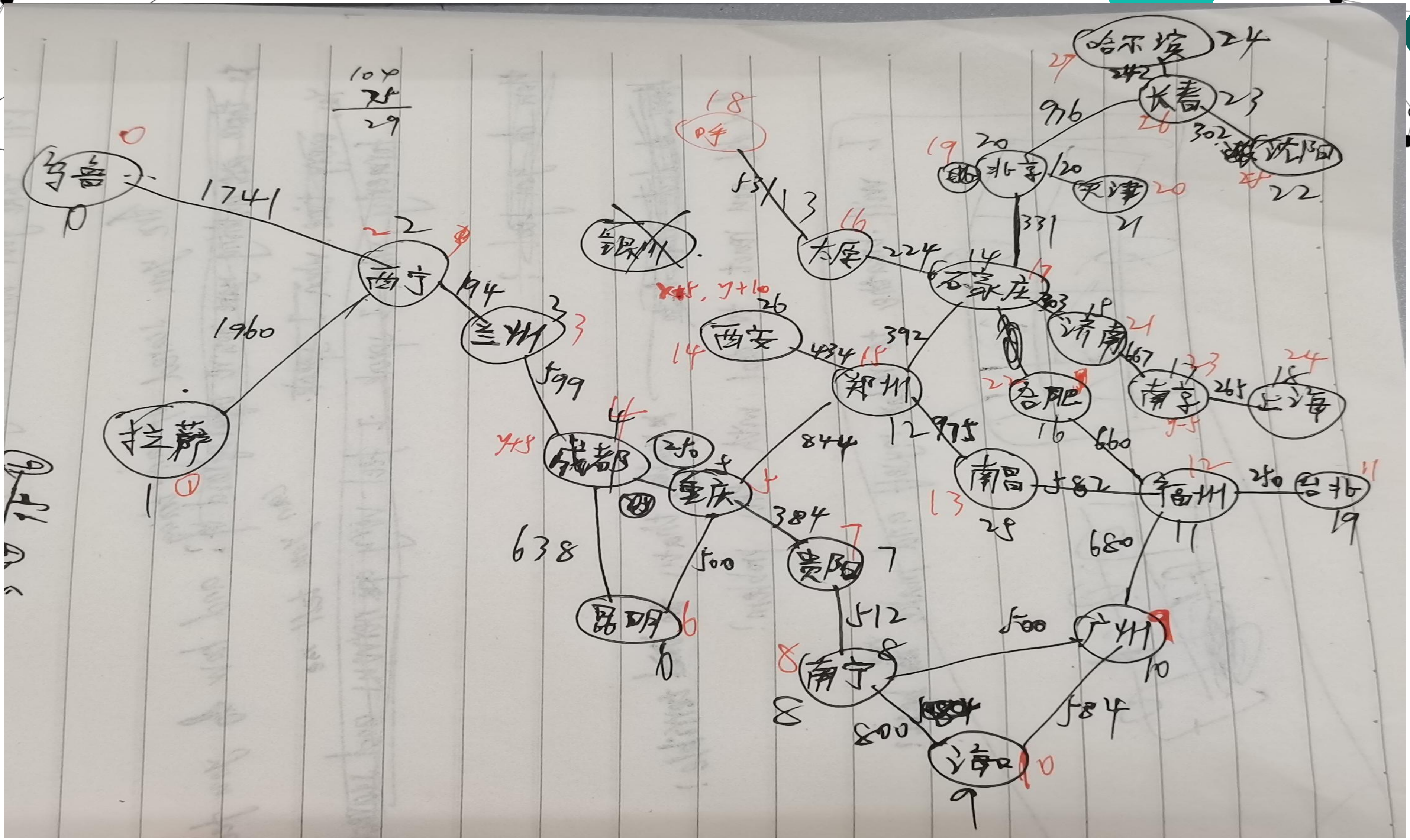

GraphicalDesign
终点：
确定
bj

城市介绍


GraphicalDesign
输入想看的城市：
确定
bj


北京

图结构

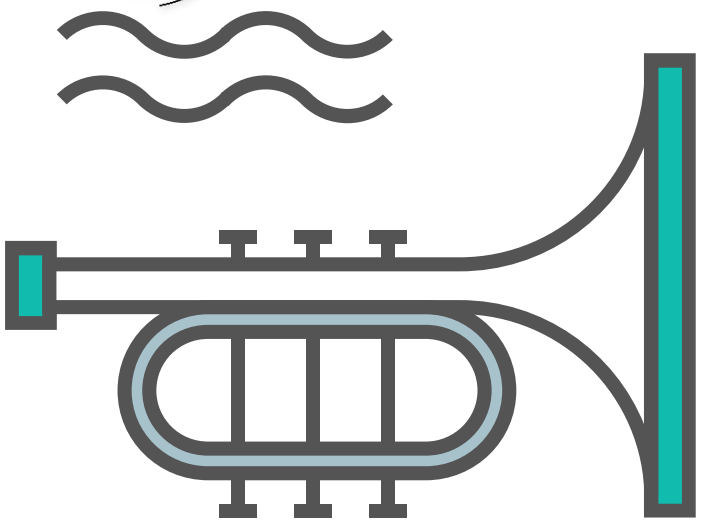


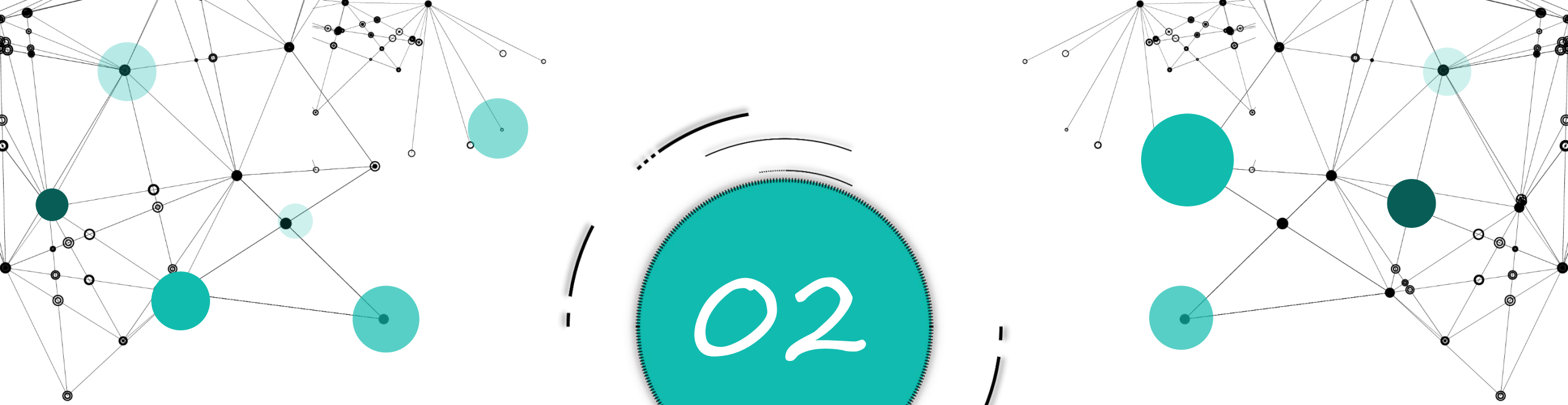
西宁
兰州
拉萨
成都
重庆
昆明
贵阳
南宁
西安
郑州
太原
石家庄
北京
天津
哈尔滨
长春
沈阳
济南
合肥
南京
上海
南昌
福州
广州
1741
1960
599
638

01

路线规划

Route planning





02

代码实现

需求分析

1.首先采用C语言的第三方库easyX库，来实现图形化界面设计。

2.本次实验室求单源最短路径，因此我们采用Dijkstra算法来求最短路径。

3.本次实验所需的图较小，直接采用邻接矩阵的方式来存储图。

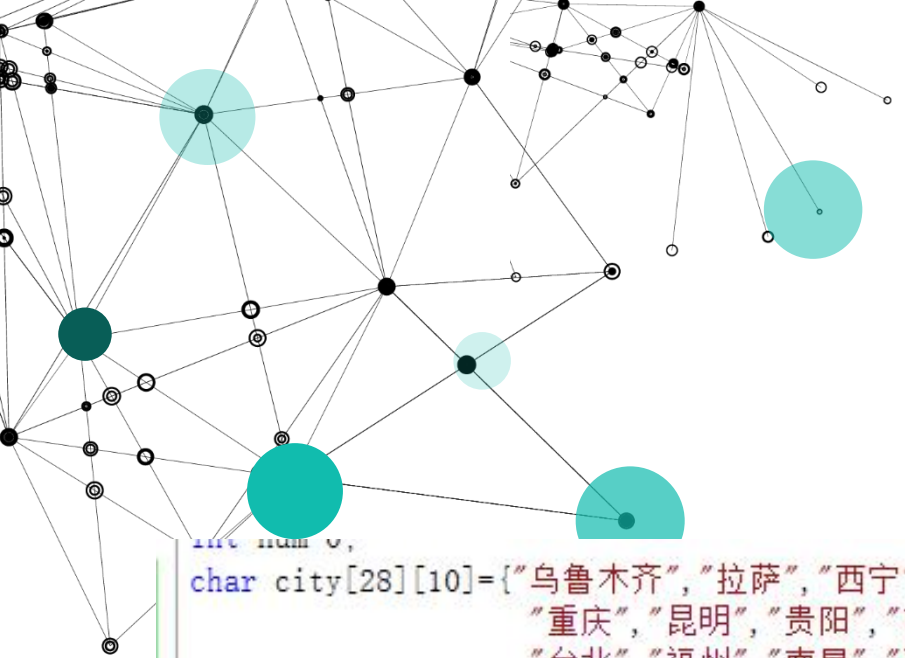
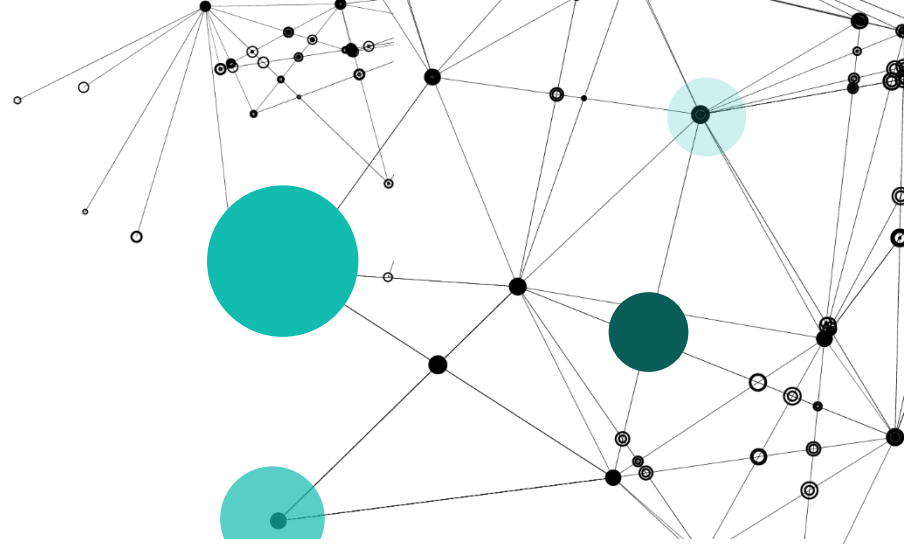
4.要实现在地图上路径的打印，需要一个定义一个path数组来存放路径。

宏定义、全局变量和头文件

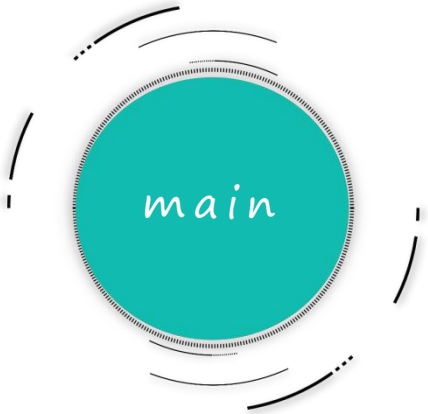
02

```
#define _CRT_SECURE_NO_WARNINGS
#include "stdafx.h"
#include <string.h>
#include<vector>
#include <conio.h>
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<graphics.h>
using namespace std;

#define N 9999
#define INF 0x3f3f3f3f
int map[N][N];  //用邻接矩阵来存图;
int dis[N];     //用了记录源点到各顶点的距离;
int vis[N];     //标记顶点是否访问过;
vector<int> path;
int p, q;//求p到q之间的最短路径;
int num=0;
```

```
int num=0;
char city[28][10]={"乌鲁木齐","拉萨","西宁","兰州","成都",
                   "重庆","昆明","贵阳","南宁","广州","海口",
                   "台北","福州","南昌","西安","郑州","太原",
                   "石家庄","呼和浩特","北京","天津","济南","合肥",
                   "南京","上海","沈阳","长春","哈尔滨"};
char city2[28][10]={"wlmq","ls","xn","lz","cd","cq","km","gy",
                   "nn","gz","hk","tb","fz","nc","xa","zz",
                   "ty","sjz","hhht","bj","tj","jn","hf",
                   "nj","sh","sy","cc","heb"};
char picture[28][10]={"wlmq.jpg","ls.jpg","xn.jpg","lz.jpg",
                   "cd.jpg","cq.jpg","km.jpg","gy.jpg","nn.jpg",
                   "gz.jpg","hk.jpg","tb.jpg","fz.jpg","nc.jpg",
                   "xa.jpg","zz.jpg","ty.jpg","sjz.jpg","hhht.jpg",
                   "bj.jpg","tj.jpg","jn.jpg","nj.jpg","sh.jpg","sy.jpg",
                   "cc.jpg","heb.jpg"};
int r[3][4]={{1010,10,1190,40},{1010,50,1190,80},{1010,640,1190,670}};//按钮坐标
int point[28][2]={{230,220},{225,520},{405,410},{445,420},{445,515},{505,545},{415,640},{485,620},{505,690},{700,750},//每个城市的坐标
                  {455,740},{735,630},{695,620},{645,570},{520,440},{595,445},{580,370},{605,370},{560,320},{625,335},
                  {645,345},{645,395},{650,495},{680,505},{720,505},{730,285},{740,235},{760,195}};
```

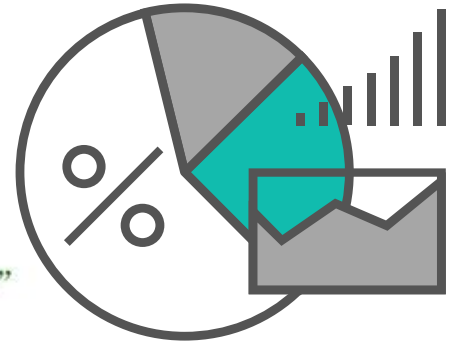



```
int p=-1,q=-1;
vector<int> P;
int start[2],end[2];
inin(30);//初始化图
initgraph(1200,1000);
IMAGE img;
IMAGE im;
loadimage(&img,"a.jpg",1000,800);//打开地图
putimage(0,0,&img);
setbkmode(TRANSPARENT);
settextstyle(30,0,"宋体");//s为字体种类如“宋体”
settextcolor(RED);
outtextxy(450,10,"导游图");
settextstyle(20,0,"宋体");

RECT R1={r[0][0],r[0][1],r[0][2],r[0][3]};
RECT R2={r[1][0],r[1][1],r[1][2],r[1][3]};
RECT R3={r[2][0],r[2][1],r[2][2],r[2][3]};
drawtext("开始导游",&R1,DT_CENTER | DT_VCENTER | DT_SINGLELINE);
drawtext("城市介绍",&R2,DT_CENTER | DT_VCENTER | DT_SINGLELINE);
drawtext("退出程序",&R3,DT_CENTER | DT_VCENTER | DT_SINGLELINE);
```

```
while(true){

    MOUSEMSG m;//鼠标指针
    m = GetMouseMsg();//获取一条鼠标消息
    switch(m.uMsg){
        case WM_LBUTTONDOWN://左键点击
            switch(button_judge(m.x,m.y)){//判断鼠标点击哪个按钮
```



case1

开始导游

```
setfillcolor(BLACK);
solidrectangle(1010,10,1190,80);//将初始信息覆盖，生成新的信息
outtextxy(1000,10,"起点：");
rectangle(1100,10, 1190, 30);//绘制边框
outtextxy(1000,40,"终点：");
rectangle(1100,40, 1190, 60);//绘制边框
char a[10],b[10];
InputBox(a,30,"起点：");//输入起点城市
InputBox(b,30,"终点：");//输入终点城市
for(int i=0;i<28;i++){
    if(strcmp(a,city2[i])==0){
        p=i;
        outtextxy(1100,10,city[p]);
    }
    if(strcmp(b,city2[i])==0){
        q=i;
        outtextxy(1100,40,city[q]);
    }
}
if(p==-1||q==-1){
    outtextxy(1100,90,"输入错误");
    system("pause");
    return 0;
}
```

```
Dijkstra(p,28);
get_path(q,P);
setcolor(YELLOW);
outtextxy(1000,70,"推荐路线:");
outtextxy(1000,110,city[p]);//起点
char t[5];
sprintf(t, "%d",dis[p]);
outtextxy(1080,110,t);
fillcircle(point[p][0],point[p][1],5);
start[0]=point[p][0];
start[1]=point[p][1];

for(int i=0;i<num;i++){//打印路径
    end[0]=point[P[i]][0];
    end[1]=point[P[i]][1];
    char s[10];
    sprintf(s, "%d",dis[P[i]]);
    setcolor(YELLOW);
    outtextxy(1000,130+i*20,city[P[i]]);
    outtextxy(1080,130+i*20,s);
    fillcircle(point[P[i]][0],point[P[i]][1],5);
    setcolor(RED);
    line(start[0],start[1],end[0],end[1]);
    start[0]=end[0];
    start[1]=end[1];
}
system("pause");
```

case2
case3

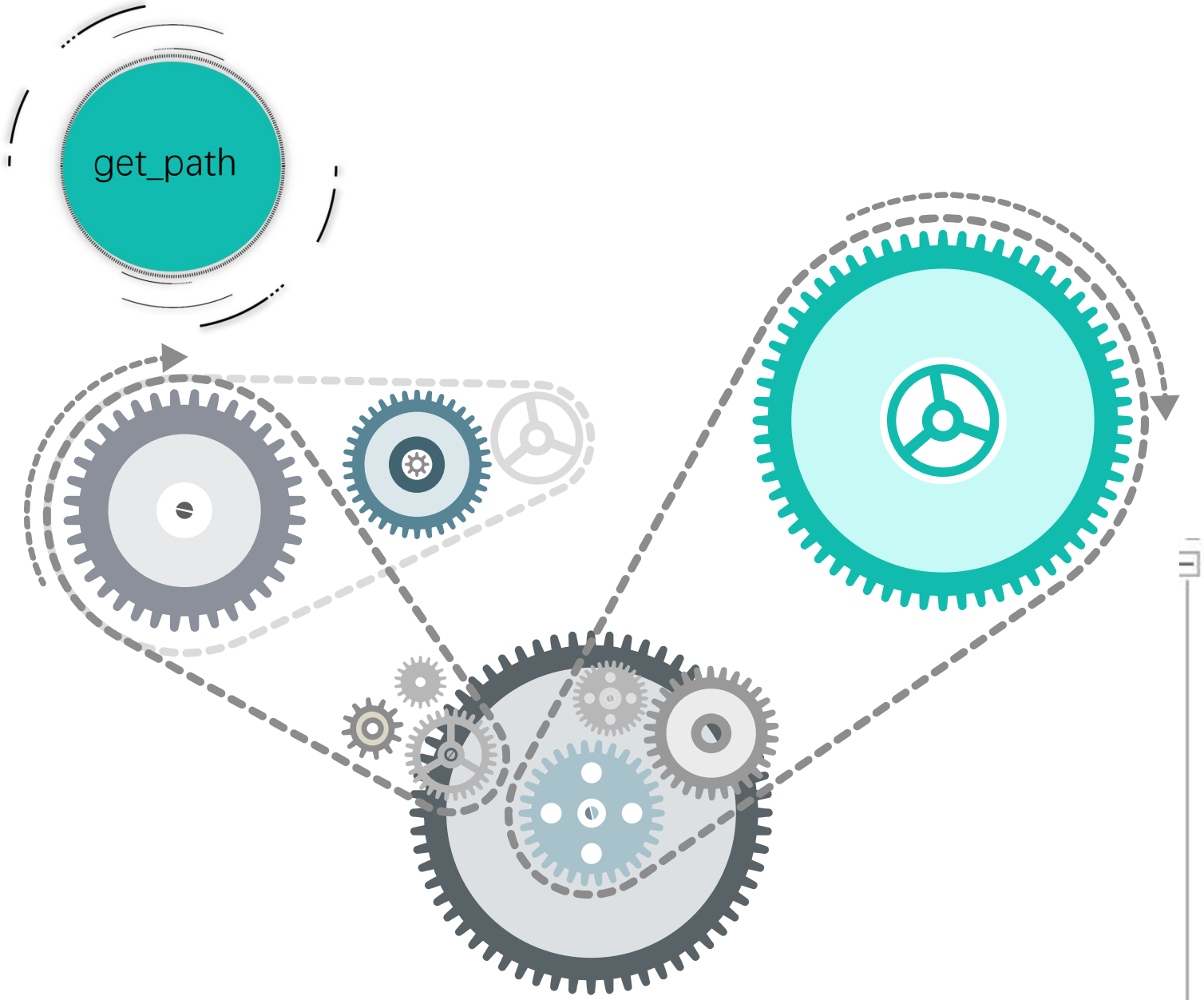
城市介绍

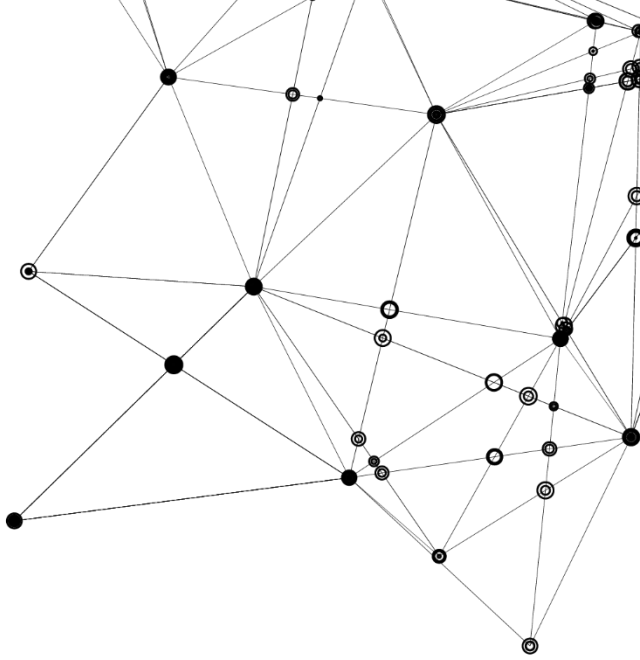
```
        setfillcolor(BLACK);
        solidrectangle(1000,360,1200,400);
        setfillcolor(RED);
        char c[10];
        InputBox(c,30,"输入想看的城市：");
        for(int i=0;i<28;i++){//查找城市
            if(strcmp(c,city2[i])==0){
                setcolor(YELLOW);
                settextstyle(14,0,"宋体");
                outtextxy(1000,380,city[i]);
                loadimage(&im,picture[i],200,200);
                putimage(1000,400,&im);
                break;
            }
        }
    getchar();
    break;
```

退出程序

Dijkstra

```
void Dijkstra(int st,int n){
    int i, j, k, x;
    path=vector<int>(n,-1);
    for (i = 0; i < n; i++)//用dis数组记录源点到与它相连接的顶点的距离；
        dis[i] = map[st][i];
    vis[st] = 1;//标记刚才源点,表示已经访问；
    while (1){
        x = -1;
        int min = INF;
        for (i = 0; i < n; i++){//在当前的dis距离数组中找到一个最小的路径，并将这条路到达的顶点记录；
            if (vis[i] != 1 && dis[i] < min){
                min = dis[i];
                x = i;
            }
        }
        vis[x] = 1;
        if (x == -1)//直到所有的顶点都已访问过，结束循环
            break;
        for (i = 0 , k=0; i < n; i++){//更新dis数组，
            if (vis[i] != 1 && dis[i] > min + map[x][i]){
                dis[i] = min + map[x][i];
                path[i]=x;
            }
        }
    }
}
```





```
void get_path(int x,vector<int>&P){
    if(x==p){
        P.push_back(0);
        return;
    }
    if(path[x]!=-1)
        get_path(path[x],P);
    P.push_back(x);
    num++;
}
```

算法优化

算法分析

```
void Dijkstra(int st,int n){
    int i, j, k, x;
    path=vector<int>(n,-1);
    for (i = 0; i < n; i++)//用dis数组记录源点到与它相连接的顶点的距离;
        dis[i] = map[st][i];
    vis[st] = 1;//标记刚才源点,表示已经访问;
    while (1){
        x = -1;
        int min = INF;
        for (i = 0; i < n; i++){//在当前的dis距离数组中找到一个最小的路径，并将这条路到达的顶点记录;
            if (vis[i] != 1 && dis[i] < min){
                min = dis[i];
                x = i;
            }
        }
        vis[x] = 1;
        if (x == -1)//直到所有的顶点都已访问过，结束循环
            break;
        for (i = 0 , k=0; i < n; i++){//更新dis数组,
            if (vis[i] != 1 && dis[i] > min + map[x][i]){
                dis[i] = min + map[x][i];
                path[i]=x;
            }
        }
    }
}
```

需要遍历n次实初始化dis数组

每一次更新dis都需要遍历依次dis数组，找到起点到第i个点的最短距离。

每次都需要遍历整个dis数组来更新起点到第i个点的距离。

每次在更新dis数组时都做了很多无意义的操作

```cpp
typedef pair<int,int> P;//fisrt:边权，second:目的顶点编号
#define MAXV 1000
#define Inf 0x3f3f//>16000
int V=28,E=30;//顶点数，边数
int s,ed;//起点，终点
int cost[MAXV][MAXV];//cost[u][v] 边(u,v)的边权
int d[MAXV];//d[i]表示顶点i到起点的最短路径
int per[MAXV];
int vis[MAXV];//vis[i]标记顶点i是否已求得到起点的最短路径 ,是: 1.否: 0

void djks(){
    d[s]=0;
    int v,u;
    priority_queue<P,vector<P>, greater<P> > pq;
    pq.push({0,s});
    while(!pq.empty()){
        P p=pq.top();
        pq.pop();
        v=p.second;
        if(d[v]<p.first) continue;//顶点v已更新最短路，取出
        vis[v]=1;//扩展顶点v
        for(u=1;u<=V;++u)//以顶点v为中间节点去更新其他点到s的最短路
            if(!vis[u]&&d[u]>d[v]+cost[v][u])
                d[u]=d[v]+cost[v][u],per[u]=v,pq.push({d[u],u});
    }
}
```

Dijkstra
堆优化

```cpp
void printPath(int v,int ed){
    if(v==ed){
        cout<<v<<" ";
        return;
    }
    printPath(per[v],ed);
    cout<<v<<" ";
}
```

AM9:00

2030.12.12

THANK YOU

感谢聆听