**算法分析与设计实验报告**

**第 三 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 李平凡 | 学号 | 201907040102 | | 班级 | 计科1905 |
| 时间 | 5.20 | 地点 | 软件大楼 | | | |
| 实验名称 | 用Dijkstra贪心算法求解单源最短路径问题 | | | | | |
| 实验目的 | 通过搜索资料，要求掌握一些基本问题和典型例题的处理方式 | | | | | |
| 实验原理 | 利用贪心法每次选择离当前点边权最小的点进行处理，每次更新从这个点到这个点可达点的最短距离，更新完毕，寻找下一个待处理的点，最后将每个点处理完得出最后最短路径。 | | | | | |
| 实验步骤 | ①首先输入点数，边数，边权等信息，然后一个数组存储所有到达该下标的最短路径信息，然后从第一个点开始处理。  ②利用dj算法每次对处理点的可达点最短路径进行更新，同时利用findmins函数找到下一个未处理的边权最小的点。  ③ 将每个点都处理后输出最后的结果。 | | | | | |
| 关键代码 | void dj(vector<vector<int> > a,int n)  {  vector<int>s(n,INF);  vector<bool>f(n,false); //f数组记录点是否被处理  f[0]=true;  for(int i=0;i<a[0].size();i++)  {  s[i]=a[0][i];  }  int j=findmins(f,s); //起点从0点开始  while(j!=0)  {  for(int i=0;i<a[j].size();i++)  {  if(a[j][i]+s[j]<s[i])  {  s[i]=a[j][i]+s[j];  }  }  f[j]=true;  j=findmins(f,s); //找到要处理的当前路径最短的点  }  file1<<"从起点到各点距离"<<endl;  cout<<"从起点到各点距离"<<endl;  for(int i=0;i<n;i++)  {  file1<<s[i]<<" ";  cout<<s[i]<<" ";  }  }  这个代码对当前边权最小的点进行处理，找到这个点可达点，同时对每个可达点进行一次比较更新这些点的最短距离。  int findmins(vector<bool> &f,vector<int> s) //寻找当前最短路径的点  {  int min=INF,index=0;  for(int i=1;i<s.size();i++)  {  if(f[i]==false&&s[i]<min) //这个点没有被处理过并且当前路径最短就可以被返回进行处理  {  min=s[i];  index=i;  }  }  return index;  }  这部分代码是用来查找当前边权最短的点，所有可达点中，找到下一个未被处理的点且路径最短然后进行处理。 | | | | | |
| 测试结果 |  | | | | | |
| 实验心得 | 这个问题最终的结果全部是无穷大，主要是因为随机数生成的不好，因为不好掌控随机数的生成，有的时候第一个点到别的点是没有距离的，同时这个算法是单源最短路径，它总是从第一个点出发到其它点寻找最短路径，所以当第一个点没有到其他点的边权，那么肯定到所有点距离都是无穷大，针对这个算法，因为它是遍历了所有点和所有边，它的复杂度是跟着两者有关系，所以最终数据测试上也是设置的边和点，最终时间开销是O(n\*m)n是点数m是边数，因为在dj函数里是一定对每个点处理，然后一共m条边需要每条边都比较一下，判断一下，所以时间复杂度是O（n\*m） | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**