1087. All Roads Lead to Rome (30)

时间限制

200 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

CHEN, Yue

Indeed there are many different tourist routes from our city to Rome. You are supposed to find your clients the route with the least cost while gaining the most happiness.

**Input Specification:**

Each input file contains one test case. For each case, the first line contains 2 positive integers N (2<=N<=200), the number of cities, and K, the total number of routes between pairs of cities; followed by the name of the starting city. The next N-1 lines each gives the name of a city and an integer that represents the happiness one can gain from that city, except the starting city. Then K lines follow, each describes a route between two cities in the format "City1 City2 Cost". Here the name of a city is a string of 3 capital English letters, and the destination is always ROM which represents Rome.

**Output Specification:**

For each test case, we are supposed to find the route with the least cost. If such a route is not unique, the one with the maximum happiness will be recommended. If such a route is still not unique, then we output the one with the maximum average happiness -- it is guaranteed by the judge that such a solution exists and is unique.

Hence in the first line of output, you must print 4 numbers: the number of different routes with the least cost, the cost, the happiness, and the average happiness (take the integer part only) of the recommended route. Then in the next line, you are supposed to print the route in the format "City1->City2->...->ROM".

**Sample Input:**

6 7 HZH

ROM 100

PKN 40

GDN 55

PRS 95

BLN 80

ROM GDN 1

BLN ROM 1

HZH PKN 1

PRS ROM 2

BLN HZH 2

PKN GDN 1

HZH PRS 1

**Sample Output:**

3 3 195 97

HZH->PRS->ROM

[提交代码](https://www.patest.cn/contests/pat-a-practise/1087)

总结：

1. 掌握了map工具的基本用法；
2. 初步了解了pair的用法；
3. 对于dijkstra编程应用的了解：

事实上是一种dp,整个是建立在一种dp思想上面的，基础思路是维护最短路，即，一个起始点到一个点的最短路，一定是min（所有联通到目标点的顶点到起始点的最短路+这些顶点到目标点的路径）

#include<iostream>

#include<string>

#include<algorithm>

#include<queue>

#include<vector>

#include<sstream>

#include<stack>

#include<map>

#include<cstring>

#include<climits>

#define MAX 205

using namespace std;

int city\_num, edge\_num;

string temp;

map<string, int>city\_code;

map<int, string>city\_code\_reverse;

map<int, vector<pair<int, int>>>edge;

int happyness\_accumulation[MAX], happyness\_incity[MAX], find\_back[MAX], routine[MAX],step[MAX],Distance[MAX],visited[MAX];

int main()

{

memset(happyness\_accumulation, 0, MAX);

memset(happyness\_incity, 0, MAX);

memset(find\_back, -1, MAX);

memset(routine, 0, MAX);

memset(step, -1, MAX);

for (int i = 0; i < MAX; i++)Distance[i] = INT\_MAX;

memset(visited, 0, MAX);

cin >> city\_num >> edge\_num;

cin >> temp;

city\_code[temp] = 0;

city\_code\_reverse[0] = temp;

for (int i = 1; i < city\_num; i++)//构建城市代号以及对应映射

{

int happiness;

cin >> temp >> happiness;

happyness\_incity[i] = happiness;

city\_code[temp] = i;

city\_code\_reverse[i] = temp;

}

for (int i = 0; i < edge\_num; i++)

{

string thisone, thatone;

int length;

cin >> thisone >> thatone>>length;

edge[city\_code[thisone]].push\_back(make\_pair(city\_code[thatone], length));

edge[city\_code[thatone]].push\_back(make\_pair(city\_code[thisone], length));

}

int pos = 0;//now\_position

Distance[0] = 0;

routine[0] = 1;

step[0] = 0;

while (1)

{

visited[pos] = 1;

if (pos == city\_code["ROM"])

break;

for (int i = 0; i < edge[pos].size(); i++)

{

int next = edge[pos][i].first;

int in\_dis = edge[pos][i].second;

if (Distance[next] > Distance[pos]+in\_dis)

{

happyness\_accumulation[next] = happyness\_accumulation[pos] + happyness\_incity[next];

Distance[next] = Distance[pos] + in\_dis;

step[next] = step[pos] + 1;

find\_back[next] = pos;

routine[next] = routine[pos];

}

else if (Distance[next] == Distance[pos] + in\_dis)

{

routine[next] += routine[pos];

if (happyness\_accumulation[next] < happyness\_accumulation[pos] + happyness\_incity[next])

{

happyness\_accumulation[next] = happyness\_accumulation[pos] + happyness\_incity[next];

step[next] = step[pos] + 1;

find\_back[next] = pos;

}

else if (happyness\_accumulation[next] == happyness\_accumulation[pos] + happyness\_incity[next])

{

if (step[next] > step[pos] + 1)

{

step[next] = step[pos] + 1;

find\_back[next] = pos;

}

}

}

}

int minipos = -1, minidis = INT\_MAX;

for (int i = 1; i < city\_num; i++)

{

if (!visited[i] && Distance[i] < minidis)

{

minipos = i;

minidis = Distance[i];

}

}

pos = minipos;

}

cout << routine[pos] << " " << Distance[pos] << " " << happyness\_accumulation[pos] << " " << happyness\_accumulation[pos] / step[pos] << endl;

stack<string>result;

while (1)

{

result.push(city\_code\_reverse[pos]);

if (find\_back[pos] == -1)

break;

pos = find\_back[pos];

}

cout << result.top();

result.pop();

while (1)

{

cout << "->";

cout << result.top();

result.pop();

if (result.empty())

break;

}

return 0;

}

1 #include<cstdio>

2 #include<cstring>

3 #include<stack>

4 #include<algorithm>

5 #include<iostream>

6 #include<stack>

7 #include<set>

8 #include<map>

9 #include<vector>

10 using namespace std;

11 map<string,int> city;//string-int//存储城市名和其值

12 map<int,string> rcity//？？;

13 map<int,vector<pair<int,int> > > edge;

14 int dis[205],path[205],hcount[205],happ[205],fstep[205],f[205];

15 bool vis[205];

16 int main()

17 {

18 //freopen("D:\\INPUT.txt","r",stdin);

19 int n,k,i,d,s;

20 string st,u,v;

21 scanf("%d %d",&n,&k);

22 memset(dis,-1,sizeof(dis));//每个点的最短路长

23 memset(hcount,-1,sizeof(hcount));//幸福指数之和

24 memset(vis,false,sizeof(vis));//是否计算过最短路径

25 memset(path,0,sizeof(path));//前一个点

26 memset(fstep,0,sizeof(fstep));//到这个点需要几步

27 cin>>st;//起始城市

28 city[st]=0;//编号

29 rcity[0]=st;//反编号

30 happ[0]=0;

31 dis[0]=0;

32 hcount[0]=0;

33 fstep[0]=0;

34 path[0]=1;//init

35 f[0]=0;//正确回溯路径

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

37 {

38 f[i]=i;

39 cin>>u;

40 rcity[i]=u;//将城市标号，然后映射到结果；

41 city[u]=i;

42 scanf("%d",&happ[i]);

43 }

44

45 /\*for(i=0;i<n;i++){

46 cout<<rcity[i]<<endl;

47 }\*/

48

49

50 for(i=0; i<k; i++)

51 {

52 cin>>u>>v;

53 scanf("%d",&d);

54 //cout<<u<<" "<<v<<" "<<d<<endl;

55 edge[city[u]].push\_back(make\_pair(city[v],d));

56

57 edge[city[v]].push\_back(make\_pair(city[u],d));//对于map进行加边操作

58 }

59

60 /\*for(i=0;i<n;i++){

61 vector<pair<int,int> >::iterator it;

62 cout<<"i: "<<i<<endl;

63 for(it=edge[i].begin(); it!=edge[i].end(); it++){

64 cout<<it->first<<" "<<it->second<<endl;

65 }

66 }\*/

67

68 s=0;

69 vector<pair<int,int> >::iterator it;

70 int next;

71 while(s!=city["ROM"])

72 {

73

74 //cout<<"s: "<<s<<endl;

75

76 vis[s]=true;

77 for(it=edge[s].begin(); it!=edge[s].end(); it++) //update

78 {

79 next=it->first;

80 if(dis[next]==-1||dis[next]>dis[s]+it->second)//没有被访问过

81 {

82 dis[next]=dis[s]+it->second;

83 hcount[next]=hcount[s]+happ[next];

84 path[next]=path[s];

85 fstep[next]=fstep[s]+1;

86 f[next]=s;

87 }

88 else

89 {

90 if(dis[next]==dis[s]+it->second)

91 {

92 path[next]+=path[s];//

93 if(hcount[next]<hcount[s]+happ[next])

94 {

95 hcount[next]=hcount[s]+happ[next];

96 fstep[next]=fstep[s]+1;

97 f[next]=s;

98 }

99 else

100 {

101 if(hcount[next]==hcount[s]+happ[next])

102 {

103 if(fstep[next]>fstep[s]+1)

104 {

105 fstep[next]=fstep[s]+1;

106 f[next]=s;

107 }

108 }

109 }

110 }

111 }

112 }

113

114 /\*for(i=1;i<n;i++){

115 cout<<"i: "<<i<<" "<<dis[i]<<endl;

116 }\*/

117

118 int mindis=-1,minnum;

119 for(i=1;i<n;i++)//find the min

120 {

121 if(dis[i]==-1){//如果当前边到不了初始点，直接pass

122 continue;

123 }

124 if(!vis[i]&&(mindis==-1||(dis[i]<mindis))){

125 //cout<<"ii: "<<i<<" "<<dis[i]<<endl;

126 mindis=dis[i];

127 minnum=i;

128 }

129 }

130

131 //cout<<"minnum: "<<minnum<<" "<<dis[minnum]<<endl;

132

133 s=minnum;

134 }

135 printf("%d %d %d %d\n",path[s],dis[s],hcount[s],hcount[s]/fstep[s]);

136 int p=s;

137 stack<int> ss;

138 while(p){

139 ss.push(p);

140 p=f[p];

141 }

142 cout<<rcity[p];

143 while(!ss.empty()){

144 cout<<"->"<<rcity[ss.top()];

145 ss.pop();

146 }

147 cout<<endl;

148 return 0;

149 }