算法

第六章 深度优先搜索算法

dfs

深度优先搜索(Depth-First-Search)

从起点出发，走过的点要做标记，发现有没走过的点，就随意挑一个往前走，走不了就回退，此种路径搜索策略就称为“深度优先搜索”，简称“深搜”。

其实称为“远度优先搜索”更容易理解些。因为这种策略能往前走一步就往前走一步，总是试图走得更远。所谓远近(或深度)，就是以距离起点的步数来衡量的。

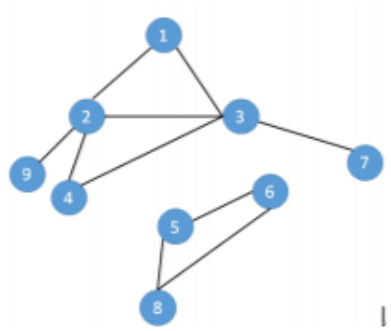
在图上寻找路径

➢判断从V出发是否能走到终点：

```
bool Dfs(V) {
        if( V 为终点)
                return true;
        if( V 为旧点)
                return false;
        将V标记为旧点;
        对和V相邻的每个节点U {
                if( Dfs(U) == true)
                        return true;
        }
        return false;
}
```

深度优先遍历图上所有节点

```
Dfs(V) {
        if( V是旧点)
                return;
        将v标记为旧点;
        对和v相邻的每个点 U {
                Dfs(U);
        }
}
int main() {
        将所有点都标记为新点;
        while(在图中能找到新点k)
                Dfs(k);
}
```



19

dfs表达

图的表示方法 -- 邻接矩阵

用一个二维数组G存放图，G[i][j]表示节点i和节点j之间边的情况(如有无边，边方向，权值大小等)。

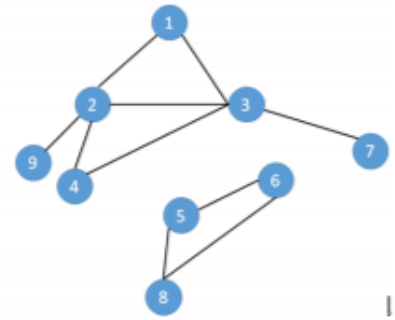
遍历复杂度：$O(n^2)$　　n为节点数目

图的表示方法 -- 邻接表

每个节点V对应一个一维数组(vector)，里面存放从V连出去的边，边的信息包括另一顶点，还可能包含边权值等。

1	2	3		
2	1	4	9	3
3	1	4	7	2
4	2	3		
5	6	8		
6	5	8		
7	3			
8	5	6		
9	2			

遍历复杂度：O(n+e)
n为节点数目，e为边数目



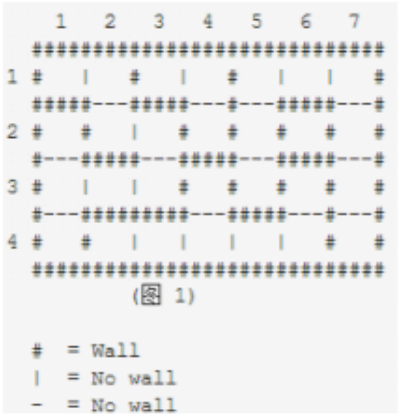
21

城堡问题

题解

例题：百练2815 城堡问题

- 右图是一个城堡的地形图。请你编写一个程序，计算城堡一共有多少房间，最大的房间有多大。城堡被分割成m×n(m≤50，n≤50)个方块，每个方块可以有0~4面墙。



输入输出

- 输入
  - 程序从标准输入设备读入数据。
  - 第一行是两个整数，分别是南北向、东西向的方块数。
  - 在接下来的输入行里，每个方块用一个数字(0≤p≤50)描述。用一个数字表示方块周围的墙，1表示西墙，2表示北墙，4表示东墙，8表示南墙。每个方块用代表其周围墙的数字之和表示。城堡的内墙被计算两次，方块(1,1)的南墙同时也是方块(2,1)的北墙。
  - 输入的数据保证城堡至少有两个房间。
- 输出
  - 城堡的房间数、城堡中最大房间所包括的方块数。
  - 结果显示在标准输出设备上。

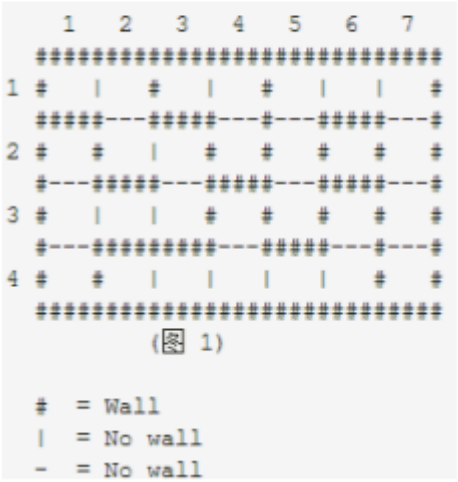
- 样例输入

4
7
11 6 11 6 3 10 6
7 9 6 13 5 15 5
1 10 12 7 13 7 5
13 11 10 8 10 12 13

- 样例输出

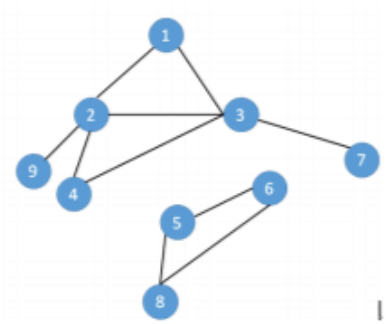
5
9

数据保证城堡
四周都是墙



解题思路

- 把方块看作是节点，相邻两个方块之间如果没有墙，则在方块之间连一条边，这样城堡就能转换成一个图。
- 求房间个数，实际上就是在求图中有多少个极大连通子图。
- 一个连通子图，往里头加任何一个图里的其他点，就会变得不连通，那么这个连通子图就是极大连通子图。（如：(8,5,6)）



code c++

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
vector<vector<int>>rooms = {
    {11,6,11,6,3,10,6},
    {7,9,6,13,5,15,5},
    {1,10,12,7,13,7,5},
    {13,11,10,8,10,12,13}
};
vector<vector<int>>colors;
int maxRoomArea = 0, roomNum = 0;
int roomArea;
void dfs(int i, int j){
    if (colors[i][j])return;
    ++roomArea;
    colors[i][j] = roomNum;
    //西北东南 rooms 某一位位0  说明 没有墙， 没有墙就继续走
    if ((rooms[i][j] & 1) == 0)dfs(i, j - 1);
    if ((rooms[i][j] & 2) == 0)dfs(i-1, j );
    if ((rooms[i][j] & 4) == 0)dfs(i, j + 1);
    if ((rooms[i][j] & 8) == 0)dfs(i+1, j );
}
void main(){
    int r = rooms.size();
    int c = rooms[0].size();
    vector<int>a(c, 0);
    colors = vector<vector<int>>(r, a);
    for (int i = 0; i < r; ++i){
        for (int j = 0; j < c; ++j){
            if (!colors[i][j]){//找完一个链接房间
                ++roomNum; roomArea = 0;
                dfs(i, j);
                maxRoomArea = max(roomArea, maxRoomArea);
            }
        }
    }
```

```
    cout << roomNum << endl;
    cout << maxRoomArea << endl;
    system("pause");

}
```

踩方格

题解

有一个方格矩阵，矩阵边界在无穷远处。我们做如下假设：

a.每走一步时，只能从当前方格移动一格，走到某个相邻的方格上；
b. 走过的格子立即塌陷无法再走第二次；
c. 只能向北、东、西三个方向走；

请问：如果允许在方格矩阵上走n步(n<=20)，共有多少种不同的方案。
2种走法只要有一步不一样，即被认为是不同的方案。

思路：

递归
从 (i,j) 出发，走n步的方案数，等于以下三项之和：

从(i+1,j)出发，走n-1步的方案数。前提：(i+1,j)还没走过
从(i,j+1)出发，走n-1步的方案数。前提：(i,j+1)还没走过
从(i,j-1)出发，走n-1步的方案数。前提：(i,j-1)还没走过

code c++

```
#include<iostream>
#include<vector>
using namespace std;
vector<int> a(50, 0);
vector<vector<int>> b(30, a);
int ways(int i, int j, int n){
    if (n == 0)return 1;
    b[i][j] = 1;
```

```
    int num = 0;
    if (!b[i][j - 1])num += ways(i, j - 1, n - 1);
    if (!b[i][j +1])num +=ways(i, j + 1, n - 1);
    if (!b[i+1][j ])num +=ways(i+1, j  , n - 1);
    b[i][j] = 0;// 保证所有方法执行 需要设为0
    return num;
}
void main(){
    int n;
    cin >> n;
    cout << ways(0, 25, n) << endl;
    system("pause");
}
```

Roads

题解

ROADS (百练1724)

N个城市，编号1到N。城市间有R条单向道路。
每条道路连接两个城市，有长度和过路费两个属性。
Bob只有K块钱，他想从城市1走到城市N。问最短共需要走多长的路。如果到不了N，输出-1

2<=N<=100
0<=K<=10000
1<=R<=10000
每条路的长度 L, 1 <= L <= 100
每条路的过路费T , 0 <= T <= 100

输入：
K
N
R
s_1 e_1 L_1 T_1
s_1 e_2 L_2 T_2
...
s_R e_R L_R T_R
s e是路起点和终点

解题思路

从城市 1开始深度优先遍历整个图，找到所有能到达 N 的走法，
选一个最优的。

最优性剪枝：

1) 如果当前已经找到的最优路径长度为L ,那么在继续搜索的过程中，总长度已经大于等于L的走法，就可以直接放弃，不用走到底了

保存中间计算结果用于最优性剪枝：

2) 用midL[k][m] 表示：走到城市k时总过路费为m的条件下，最优路径的长度。若在后续的搜索中，再次走到k时，如果总路费恰好为m，且此时的路径长度已经超过midL[k][m],则不必再走下去了。

code c++

```
#include<iostream>
#include<vector>
#include<algorithm>
```

```
using namespace std;
int K, N, R;//k 有k元 N终点城市N 城市间有R条单向道路
struct Road{
    int d, L, t;
    //d 从当前路走到终点城市d需要的路长L和过路费t
};
vector<vector<Road>>G(110);
int minL[110][10010];//minl[i][j]代表从起点1走到了城市i 所花费的钱j 所走的路长
//G[i] 存放从i开始的边 Road 所以只需要记录终点d
int minLen;//最佳终点路径的长度
int totalLen;// 正在看的这条路走了多长
int totalCost;//正在走的这条路花了多少钱
int visited[110];//记录城市有没有走过

//会超时 交到oj上 需要剪枝 这样时间才能通过
void dfs(int s){
    if (s == N){
        minLen = min(totalLen, minLen);
        return;
    }
    for (int i = 0; i < G[s].size(); ++i){
        Road r = G[s][i];
        if (totalCost + r.t>K)continue;//可行性剪枝
        if (!visited[r.d]){
            if (totalLen + r.L >= minLen)continue;//最优性剪纸
            if (totalLen + r.L >= minL[r.d][totalCost + r.t])
                continue;//最优性剪纸2
            minL[r.d][totalCost + r.t] = totalLen + r.L;

            totalLen += r.L;
            totalCost += r.t;
            visited[r.d] = 1;
            for (int i = 0; i < 110;++i)
                for (int j = 0; j < 10010l; ++j)
                    minL[i][j] = INT_MAX;
            dfs(r.d);
            // dfs(r.d)这条路试玩以后 从s开始的要实验下一条路
            //因为下一条路 可能再走到d上，所以要删除这些标志信息
            visited[r.d] = 0;
            totalLen -= r.L;
            totalCost -= r.t;
            //进入下一个路
        }

    }

}
int main(){
    //读数据
    cin >> K >> N >> R;
    for (int i = 0; i < R; ++i){
        int s; Road r;
        cin >> s >> r.d >> r.L >> r.t;
        if (s != r.d)G[s].push_back(r);
    }
    memset(visited, 0, sizeof(visited));
    totalLen = 0;
    minLen = INT_MAX;
```

```
    totalCost = 0;
    visited[1] = 1;
    dfs(1);
    if (minLen < INT_MAX)cout << minLen << endl;
    else cout << "-1" << endl;
    system("pause");
    return 0;
}
```