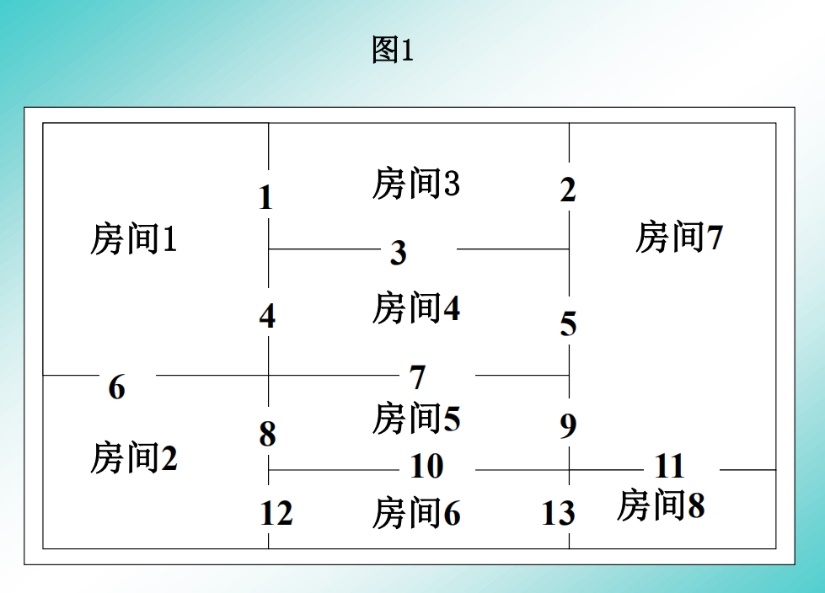
1. 某艺术馆考虑安装-一个摄像安全系统以减少其保安费用，图1是该艺术馆用以展览的房间示意图，房间的通道显示为1- -13。一家保安公司建议在一些通道安装双向摄像机，每架双向摄像机都可以监视到其两侧的房间，如，在通道4安装摄像机，房间1和房间4就可以被监视，在通道11处安装摄像机，房间7和房间8就可以被监视。管理部门决定不在入口处安装摄像机，请给出双向摄像机使用数量最少而能覆盖所有8间房的摄像机安装方案。若房间7的陈列品尤为重要，要求两架摄像机监视该房间，请给出双向摄像机安装的数量及位置。



根据题目，可以把上图抽象成如下

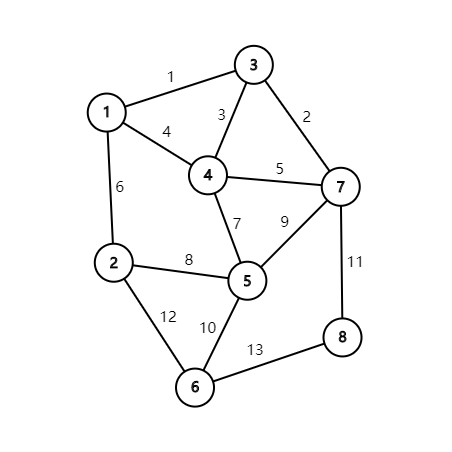


图1.1

每个房间代表一个节点，每个通道代表连接两两房间的边，而边上的权值为通道的号码。图1.1为原题目所得的无向图。

符号说明

|  |  |
| --- | --- |
| 符号 | 说明 |
|  | 图5.1的邻接矩阵 |
|  | 房间与房间之间是否有通道 |
|  | 所有房间节点的集合 |
| E | 房间通道的集合 |
|  | 布置摄像机的邻接矩阵 |
|  | 摄像机 |

构造如下的邻接矩阵

由题意可知，每个节点的度

显然此时

可以列出以下方程

根据深度优先搜索思路，可以将这个方程组进行简化，根据程序直接求出所有满足条件的

解得共有6种方案，最小摄像机数量为4。

分别为

|  |  |  |
| --- | --- | --- |
| 房间号 | 房间号 | 通道号 |
| 1 | **2** | **6** |
| 4 | **5** | **7** |
| 3 | **7** | **2** |
| 8 | **6** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 7 | **8** | **11** |
| 3 | **4** | **3** |
| 5 | **6** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 6 | **8** | **13** |
| 7 | **5** | **9** |
| 4 | **3** | **3** |
| 房间号 | **房间号** | **通道号** |
| 1 | **3** | **1** |
| 4 | **5** | **7** |
| 2 | **6** | **12** |
| 8 | **7** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **3** | **1** |
| 4 | **7** | **5** |
| 5 | **2** | **8** |
| 8 | **6** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **4** | **4** |
| 3 | **7** | **2** |
| 5 | **2** | **8** |
| 8 | **6** | **13** |

若房间7需要两架摄像机在以上最优的情况下，再增加一个摄像机。故此时最少需要5架摄像机，满足方案的有18种，如下

|  |  |  |
| --- | --- | --- |
| 房间号 | 房间号 | 通道号 |
| 1 | **2** | **6** |
| 3 | **7** | **2** |
| 4 | **5** | **7** |
| 4 | **7** | **5** |
| 6 | **8** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 3 | **7** | **2** |
| 4 | **5** | **7** |
| 5 | **7** | **9** |
| 6 | **8** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 3 | **7** | **2** |
| 4 | **5** | **7** |
| 6 | **8** | **13** |
| 7 | **8** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 3 | **4** | **3** |
| 3 | **7** | **2** |
| 5 | **6** | **10** |
| 7 | **8** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 3 | **4** | **3** |
| 4 | **7** | **5** |
| 5 | **6** | **10** |
| 7 | **8** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 3 | **4** | **3** |
| 5 | **6** | **10** |
| 5 | **7** | **9** |
| 7 | **8** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 3 | **4** | **3** |
| 3 | **7** | **2** |
| 5 | **7** | **9** |
| 6 | **8** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 3 | **4** | **3** |
| 4 | **7** | **5** |
| 5 | **7** | **9** |
| 6 | **8** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **2** | **6** |
| 3 | **4** | **7** |
| 5 | **7** | **12** |
| 6 | **8** | **13** |
| 7 | **8** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **3** | **1** |
| 2 | **6** | **12** |
| 3 | **7** | **2** |
| 4 | **5** | **7** |
| 7 | **8** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **3** | **1** |
| 2 | **6** | **12** |
| 4 | **5** | **7** |
| 4 | **7** | **5** |
| 7 | **8** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **3** | **1** |
| 2 | **6** | **12** |
| 4 | **5** | **7** |
| 5 | **7** | **9** |
| 7 | **8** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **3** | **1** |
| 2 | **5** | **8** |
| 3 | **7** | **2** |
| 4 | **7** | **5** |
| 6 | **8** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **3** | **1** |
| 2 | **5** | **8** |
| 4 | **7** | **5** |
| 5 | **7** | **9** |
| 6 | **8** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **3** | **1** |
| 2 | **5** | **8** |
| 4 | **7** | **5** |
| 6 | **8** | **13** |
| 7 | **8** | **11** |
| 房间号 | **房间号** | **通道号** |
| 1 | **4** | **4** |
| 2 | **5** | **8** |
| 3 | **7** | **2** |
| 4 | **7** | **5** |
| 6 | **8** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **4** | **4** |
| 2 | **5** | **8** |
| 3 | **7** | **2** |
| 5 | **7** | **9** |
| 6 | **8** | **13** |
| 房间号 | **房间号** | **通道号** |
| 1 | **4** | **1** |
| 2 | **5** | **8** |
| 3 | **7** | **2** |
| 6 | **8** | **13** |
| 7 | **8** | **11** |

附录

源代码

**#include <iostream>**

**#include <vector>**

**#include <cstdio>**

**#define size\_mp 8**

**using namespace std;**

**vector<int>G[10];**

**int degree[10],idx,dfn;**

**bool vis[10];**

**int track[10000][2],history[10000][8][8];**

**void dfs(int v){**

**if(vis[v]==true){**

**for (auto e : G[v]){**

**if(degree[e]>1 && vis[e]==false){**

**dfs(e);**

**}**

**}**

**}**

**else**

**{**

**for (auto e : G[v]){**

**if(degree[e]>1 && vis[e]==false){**

**degree[e]--;**

**degree[v]--;**

**vis[v] = true;**

**vis[e] = true;**

**dfn++;**

**track[dfn][0] = v;**

**track[dfn][1] = e;**

**dfs(e);**

**vis[e] = false;**

**vis[v] = false;**

**track[dfn][0] = 0;**

**track[dfn][1] = 0;**

**dfn--;**

**degree[e]++;**

**degree[v]++;**

**}**

**}**

**}**

**int cnt = 0;**

**for(int i=1;i<=size\_mp;i++){**

**if(vis[i] == true) cnt++;**

**}**

**if(cnt == 8){**

**for(int k=1;k<=idx;k++){**

**int flag = 0;**

**for(int i=1;i<=dfn;i++){**

**if(history[k][track[i][0]-1][track[i][1]-1] != 0){**

**flag++;**

**}**

**if(flag == dfn) goto here;**

**}**

**}**

**idx++;**

**if(idx == 4){**

**cout<<"hello world"<<endl;**

**}**

**for(int i=1;i<=dfn;i++){**

**cout<<track[i][0]<<"<----->"<<track[i][1]<<endl;**

**history[idx][track[i][0]-1][track[i][1]-1] = 1;**

**history[idx][track[i][1]-1][track[i][0]-1] = 1;**

**}**

**printf("-----------condition%d---------\n",idx);**

**}**

**here:{}**

**}**

**bool check(int x,int y,int m,int maxidx){**

**int upperlimit = 10,cnt = 0;**

**history[m][x][y] = 1;**

**history[m][y][x] = 1;**

**for(int k=idx+1;k<=maxidx;k++){**

**for(int i=0;i<size\_mp;i++){**

**for(int j=0;j<size\_mp;j++){**

**if(history[k][i][j]==history[m][i][j]) cnt++;**

**if(cnt>=upperlimit) return false;**

**}**

**}**

**}**

**history[m][x][y] = 0;**

**history[m][y][x] = 0;**

**return true;**

**}**

**void copymp(int m,int x,int y){**

**for(int i=0;i<size\_mp;i++){**

**for(int j=0;j<size\_mp;j++){**

**history[idx][i][j] = history[m][i][j];**

**}**

**}**

**history[idx][x][y] = 1;**

**history[idx][y][x] = 1;**

**}**

**int main(){**

**int mp[8][8]={**

**{0,1,1,1,0,0,0,0},**

**{1,0,0,0,1,1,0,0},**

**{1,0,0,1,0,0,1,0},**

**{1,0,1,0,1,0,1,0},**

**{0,1,0,1,0,1,1,0},**

**{0,1,0,0,1,0,0,1},**

**{0,0,1,1,1,0,0,1},**

**{0,0,0,0,0,1,1,0}**

**};**

**for(int i=0;i<size\_mp;i++){**

**for(int j=0;j<size\_mp;j++){**

**if(mp[i][j]==1){**

**G[i+1].push\_back(j+1);**

**degree[i+1]++;**

**}**

**}**

**}**

**dfs(1);**

**int tail = idx;**

**for(int k=1;k<=tail;k++){**

**for (auto e : G[7]){**

**if(history[k][6][e-1] == 0 && check(6,e-1,k,idx)){**

**idx++;**

**if(idx == 16){**

**cout<<"hello world"<<endl;**

**}**

**copymp(k,6,e-1);**

**}**

**}**

**}**

**for(int k=tail+1;k<=idx;k++){**

**if(k == 16){**

**cout<<"hello world"<<endl;**

**}**

**for(int i=0;i<size\_mp;i++){**

**for(int j=i+1;j<size\_mp;j++){**

**if(history[k][i][j]!=0){**

**cout<<i+1<<"<------>"<<j+1<<endl;**

**}**

**}**

**}**

**cout<<"-----condition"<<k<<"------"<<endl;**

**}**

**cout<<"hello world"<<endl;**

**return 0;**

**}**