Powered by:**NEFU AB_IN**

KM

- ## 介绍

  二分图最大权分配

- ## 博客

  •学习：https://www.cnblogs.com/logosG/p/logos.html

  •代码：https://www.cnblogs.com/wenruo/p/5264235.html

- ## 板子

  用邻接矩阵写就行，毕竟这是一个$O(n^3)$算法，数据量不会太大

  注意

  - 不用重新定义n
  - 每次跑前初始化w，分要求初始化——INF or 0

```
namespace KM{
    const int N = 1e4 + 10;
    int n, ex_L[N], ex_R[N], match[N], slack[N], w[N][N];
    bool vis_L[N], vis_R[N];
    const int INF = 0x3f3f3f3f;

    bool dfs(int u){
        vis_L[u] = true;
        for(int v = 1; v <= n; ++ v){
            if(vis_R[v]) continue;
            int gap = ex_L[u] + ex_R[v] - w[u][v];
            if(!gap){
                vis_R[v] = true;
                if(!match[v] || dfs(match[v])){
                    match[v] = u;
                    return true;
                }
            }
            else{
                slack[v] = min(slack[v], gap);
            }
        }
        return false;
    }

    int km(){
```

```
        memset(match, 0, sizeof match);
        memset(ex_R, 0, sizeof ex_R);
        for(int i = 1; i <= n; ++ i){
            ex_L[i] = w[i][1];
            for(int j = 1; j <= n; ++ j){
                ex_L[i] = max(ex_L[i], w[i][j]);
            }
        }
        for(int i = 1; i <= n; ++ i){
            memset(slack, 0x3f, sizeof slack);
            while(true){
                memset(vis_L, false, sizeof vis_L);
                memset(vis_R, false, sizeof vis_R);
                if(dfs(i)) break;
                int d = INF;
                for(int j = 1; j <= n; ++ j)
                    if (!vis_R[j]) d = min(d, slack[j]);
                for(int j = 1; j <= n; ++ j){
                    if(vis_L[j]) ex_L[j] -= d;
                    if(vis_R[j]) ex_R[j] += d;
                    else slack[j] -= d;
                }
            }
        }
        int ans = 0;
        for(int i = 1; i <= n; ++ i){
            if(match[i]) ans += w[match[i]][i];
        }
        return ans;
    }
}
using namespace KM;
```

- 关于最小权值匹配

  对每条边取相反数，然后得到的结果再取相反数，就能得到最小权匹配

  例题：J

A - 奔小康赚大钱(hdu 2255)

板子题

```
/*
 * @Author: NEFU AB_IN
 * @Date: 2021-08-18 10:48:10
 * @FilePath: \Vscode\ACM\Project\KM\hdu2255.cpp
 * @LastEditTime: 2021-08-18 17:12:32
 */
#include<bits/stdc++.h>
using namespace std;
```

```
#define LL                          long long
#define ULL                         unsigned long long
#define MP                          make_pair
#define SZ(X)                       ((int)(X).size())
#define IOS
ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define DEBUG(X)                    cout << #X << ": " << X << endl;
typedef pair<int , int>             PII;

namespace KM{
    const int N = 1e4 + 10;
    int n, ex_L[N], ex_R[N], match[N], slack[N], w[N][N];
    bool vis_L[N], vis_R[N];
    const int INF = 0x3f3f3f3f;

    bool dfs(int u){
        vis_L[u] = true;
        for(int v = 1; v <= n; ++ v){
            if(vis_R[v]) continue;
            int gap = ex_L[u] + ex_R[v] - w[u][v];
            if(!gap){
                vis_R[v] = true;
                if(!match[v] || dfs(match[v])){
                    match[v] = u;
                    return true;
                }
            }
            else{
                slack[v] = min(slack[v], gap);
            }
        }
        return false;
    }

    int km(){
        memset(match, 0, sizeof match);
        memset(ex_R, 0, sizeof ex_R);
        for(int i = 1; i <= n; ++ i){
            ex_L[i] = w[i][1];
            for(int j = 1; j <= n; ++ j){
                ex_L[i] = max(ex_L[i], w[i][j]);
            }
        }
        for(int i = 1; i <= n; ++ i){
            memset(slack, 0x3f, sizeof slack);
            while(true){
                memset(vis_L, false, sizeof vis_L);
                memset(vis_R, false, sizeof vis_R);
                if(dfs(i)) break;
                int d = INF;
                for(int j = 1; j <= n; ++ j)
                    if (!vis_R[j]) d = min(d, slack[j]);
                for(int j = 1; j <= n; ++ j){
                    if(vis_L[j]) ex_L[j] -= d;
```

```
                    if(vis_R[j]) ex_R[j] += d;
                    else slack[j] -= d;
                }
            }
        }
        int ans = 0;
        for(int i = 1; i <= n; ++ i){
            if(match[i]) ans += w[match[i]][i];
        }
        return ans;
    }
}
using namespace KM;

int main()
{
    while(~scanf("%d", &n)){
        for(int i = 1; i <= n; ++ i){
            for(int j = 1; j <= n; ++ j){
                scanf("%d", &w[i][j]);
            }
        }
        printf("%d\n", km());
    }
    return 0;
}
```

B - Tour(hdu 3488)

- 题意

  给出n个点m条单向边边以及经过每条边的费用，让你求出走过一个哈密顿环（除起点外，每个点只能走一次）的最小费用。题目保证至少存在一个环满足条件。

- 思路

  - 任意类似的**有向环最小权值覆盖**问题，都可以用**最小费用流**来写

    网络的最大流等于最小割

    所以直接把城市拆点即可，然后城市到城市之间流量为INF，跑最大流即可。注意初末两个城市之间不能割

  - 没有自环，那么只要保证每个点只有一个出度和入度就可以了。把一个点拆成入点和出点，结合题目要求，便是一个完美的二分图。把边权设置为负数，跑一遍KM即可

  注意有重边，那么对于**最小权匹配**来说，就把邻接矩阵赋INF，然后对每条边求最大值

```
/*
 * @Author: NEFU AB_IN
```

```
 * @Date: 2021-08-19 16:06:12
 * @FilePath: \Vscode\ACM\Project\KM\Tour.cpp
 * @LastEditTime: 2021-08-19 16:12:01
 */
#include<bits/stdc++.h>
using namespace std;
#define LL                          long long
#define MP                          make_pair
#define SZ(X)                       ((int)(X).size())
#define IOS
ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define DEBUG(X)                    cout << #X << ": " << X << endl;
typedef pair<int , int>             PII;

namespace KM{
    const int N = 1e4 + 10;
    int n, ex_L[N], ex_R[N], match[N], slack[N], w[N][N];
    bool vis_L[N], vis_R[N];
    const int INF = 0x3f3f3f3f;

    bool dfs(int u){
        vis_L[u] = true;
        for(int v = 1; v <= n; ++ v){
            if(vis_R[v]) continue;
            int gap = ex_L[u] + ex_R[v] - w[u][v];
            if(!gap){
                vis_R[v] = true;
                if(!match[v] || dfs(match[v])){
                    match[v] = u;
                    return true;
                }
            }
            else{
                slack[v] = min(slack[v], gap);
            }
        }
        return false;
    }

    int km(){
        memset(match, 0, sizeof match);
        memset(ex_R, 0, sizeof ex_R);
        for(int i = 1; i <= n; ++ i){
            ex_L[i] = w[i][1];
            for(int j = 1; j <= n; ++ j){
                ex_L[i] = max(ex_L[i], w[i][j]);
            }
        }
        for(int i = 1; i <= n; ++ i){
            memset(slack, 0x3f, sizeof slack);
            while(true){
                memset(vis_L, false, sizeof vis_L);
                memset(vis_R, false, sizeof vis_R);
                if(dfs(i)) break;
```

```
                int d = INF;
                for(int j = 1; j <= n; ++ j)
                    if (!vis_R[j]) d = min(d, slack[j]);
                for(int j = 1; j <= n; ++ j){
                    if(vis_L[j]) ex_L[j] -= d;
                    if(vis_R[j]) ex_R[j] += d;
                    else slack[j] -= d;
                }
            }
        }
        int ans = 0;
        for(int i = 1; i <= n; ++ i){
            if(match[i]) ans += w[match[i]][i];
        }
        return ans;
    }
}
using namespace KM;

signed main()
{
    IOS;
    int t;
    cin >> t;
    while(t --){
        int m;
        cin >> n >> m;
        for(int i = 1; i <= n; ++ i)
            for(int j = 1; j <= n; ++ j)
                w[i][j] = -INF;
        for(int i = 1; i <= m; ++ i){
            int u, v, x;
            cin >> u >> v >> x;
            w[u][v] = max(w[u][v], -x);
        }
        cout << -km() << '\n';
    }
    return 0;
}
```

C - Cyclic Tour

和上题一个思路，只是多了一句判断，如果两点若匹配是INF，那么说明二分图是不成立的

```
/*
 * @Author: NEFU AB_IN
 * @Date: 2021-08-19 16:16:04
 * @FilePath: \Vscode\ACM\Project\KM\CyclicTour.cpp
 * @LastEditTime: 2021-08-19 16:25:27
 */
```

```cpp
#include<bits/stdc++.h>
using namespace std;
#define LL                          long long
#define MP                          make_pair
#define SZ(X)                       ((int)(X).size())
#define IOS
ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define DEBUG(X)                    cout << #X << ": " << X << endl;
typedef pair<int , int>             PII;

namespace KM{
    const int N = 1e4 + 10;
    int n, ex_L[N], ex_R[N], match[N], slack[N], w[N][N];
    bool vis_L[N], vis_R[N];
    const int INF = 0x3f3f3f3f;

    bool dfs(int u){
        vis_L[u] = true;
        for(int v = 1; v <= n; ++ v){
            if(vis_R[v]) continue;
            int gap = ex_L[u] + ex_R[v] - w[u][v];
            if(!gap){
                vis_R[v] = true;
                if(!match[v] || dfs(match[v])){
                    match[v] = u;
                    return true;
                }
            }
            else{
                slack[v] = min(slack[v], gap);
            }
        }
        return false;
    }

    int km(){
        memset(match, 0, sizeof match);
        memset(ex_R, 0, sizeof ex_R);
        for(int i = 1; i <= n; ++ i){
            ex_L[i] = w[i][1];
            for(int j = 1; j <= n; ++ j){
                ex_L[i] = max(ex_L[i], w[i][j]);
            }
        }
        for(int i = 1; i <= n; ++ i){
            memset(slack, 0x3f, sizeof slack);
            while(true){
                memset(vis_L, false, sizeof vis_L);
                memset(vis_R, false, sizeof vis_R);
                if(dfs(i)) break;
                int d = INF;
                for(int j = 1; j <= n; ++ j)
                    if (!vis_R[j]) d = min(d, slack[j]);
                for(int j = 1; j <= n; ++ j){
```

```
                    if(vis_L[j]) ex_L[j] -= d;
                    if(vis_R[j]) ex_R[j] += d;
                    else slack[j] -= d;
                }
            }
        }
        int ans = 0;
        for(int i = 1; i <= n; ++ i){
            if(w[match[i]][i] == -INF) return 1;
            if(match[i]) ans += w[match[i]][i];
        }
        return ans;
    }
}
using namespace KM;

signed main()
{
    IOS;
    int m;
    while(cin >> n >> m){
        for(int i = 1; i <= n; ++ i)
            for(int j = 1; j <= n; ++ j)
                w[i][j] = -INF;
        for(int i = 1; i <= m; ++ i){
            int u, v, x;
            cin >> u >> v >> x;
            w[u][v] = max(w[u][v], -x);
        }
        cout << -km() << '\n';
    }
    return 0;
}
```

D - Going Home

- 题意

  在一个$n*m$的二维平面上有x个房子，x个人，要把x个人安排到x房子里，一个人可以向四周移动，每移动一步花费一块钱，求将所有人安排完成后，一共最少花多少钱

- 思路

  还是二分图最小权匹配，主要是建图

```
/*
 * @Author: NEFU AB_IN
 * @Date: 2021-08-19 16:37:34
 * @FilePath: \Vscode\ACM\Project\KM\GoingHome.cpp
 * @LastEditTime: 2021-08-19 16:47:57
```

```
 */
#include<bits/stdc++.h>
using namespace std;
#define LL                          long long
#define MP                          make_pair
#define SZ(X)                       ((int)(X).size())
#define IOS
ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define DEBUG(X)                    cout << #X << ": " << X << endl;
typedef pair<int , int>             PII;

namespace KM{
    const int N = 110;
    int n, ex_L[N], ex_R[N], match[N], slack[N], w[N][N];
    bool vis_L[N], vis_R[N];
    const int INF = 0x3f3f3f3f;

    bool dfs(int u){
        vis_L[u] = true;
        for(int v = 1; v <= n; ++ v){
            if(vis_R[v]) continue;
            int gap = ex_L[u] + ex_R[v] - w[u][v];
            if(!gap){
                vis_R[v] = true;
                if(!match[v] || dfs(match[v])){
                    match[v] = u;
                    return true;
                }
            }
            else{
                slack[v] = min(slack[v], gap);
            }
        }
        return false;
    }

    int km(){
        memset(match, 0, sizeof match);
        memset(ex_R, 0, sizeof ex_R);
        for(int i = 1; i <= n; ++ i){
            ex_L[i] = w[i][1];
            for(int j = 1; j <= n; ++ j){
                ex_L[i] = max(ex_L[i], w[i][j]);
            }
        }
        for(int i = 1; i <= n; ++ i){
            memset(slack, 0x3f, sizeof slack);
            while(true){
                memset(vis_L, false, sizeof vis_L);
                memset(vis_R, false, sizeof vis_R);
                if(dfs(i)) break;
                int d = INF;
                for(int j = 1; j <= n; ++ j)
                    if (!vis_R[j]) d = min(d, slack[j]);
```

```
                for(int j = 1; j <= n; ++ j){
                    if(vis_L[j]) ex_L[j] -= d;
                    if(vis_R[j]) ex_R[j] += d;
                    else slack[j] -= d;
                }
            }
        }
        int ans = 0;
        for(int i = 1; i <= n; ++ i){
            if(match[i]) ans += w[match[i]][i];
        }
        return ans;
    }
}
using namespace KM;


signed main()
{
    IOS;
    int m;
    while(cin >> n >> m){
        memset(w, 0, sizeof w);
        if(!n && !m) break;
        vector <PII> mm;
        vector <PII> hh;
        for(int i = 1; i <= n; ++ i){
            for(int j = 1; j <= m; ++ j){
                char c;
                cin >> c;
                if(c == 'm') mm.emplace_back(MP(i, j));
                if(c == 'H') hh.emplace_back(MP(i, j));
            }
        }
        int i = 0;
        for(auto &m : mm){
            ++ i;
            int j = 0;
            for(auto &h : hh){
                w[i][++ j] -= abs(m.first - h.first) + abs(m.second -
h.second);
            }
        }
        n = i;
        cout << -km() << '\n';
    }
    return 0;
}
```