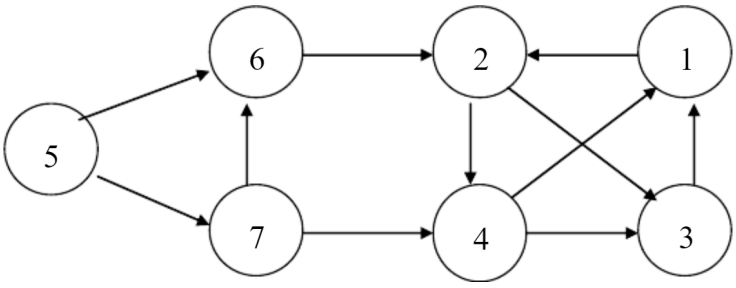
04/24/18 10:58:41 D:\git-repos\data-structure-homework\07\e20.cpp

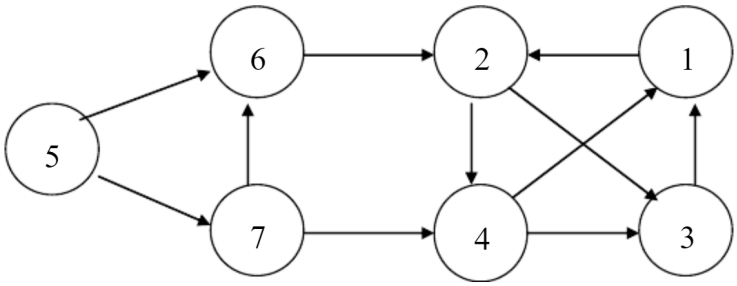
```cpp
#include <cstdio>
#include <cstring>
#include <vector>
#define MXN 107
using namespace std;
vector<int> v[MXN];
bool tag[MXN];
int n, m;
int x;
bool dfs(int root, int dep, int target, int k) {
    if (dep > k) return false;
    if (target == root && dep == k) return true;
    for (auto i : v[root]) {
        if (!tag[i]) {
            tag[i] = true;
            if (dfs(i, dep + 1, target, k))
                return true;
            tag[i] = false;
        }
    }
    return false;
}
int main() {
    scanf("%d %d", &n, &m);
    for (int i = 0; i < m; ++i) {
        int a, b;
        scanf("%d %d", &a, &b);
        v[a].push_back(b);
        v[b].push_back(a);
    }
    int a, b, k;
    while (~scanf("%d %d %d", &a, &b, &k)) {
        memset(tag, 0, sizeof(tag));
        tag[a] = true;
        puts(dfs(a, 0, b, k) ? "YES" : "NO");
    }
}
/**
 root ▶ … ≻ git-repos ≻ data-structure-homework ≻ 07 ▶ g++ e20.cpp -std=c++11
 root ▶ … ≻ git-repos ≻ data-structure-homework ≻ 07 ▶ ./a.out
7
11
5 6
5 7
7 6
6 2
7 4
2 4
2 3
4 1
1 2
3 1
4 3
1 4 5
YES
1 4 6
YES
1 4 7
NO
6 4 5
NO
6 4 3
YES
6 4 4
YES
^Z
[1]  + 1112 suspended  ./a.out
 root ▶ … ≻ git-repos ≻ data-structure-homework ≻ 07 ▶ 1 ▶    ◀ 148 ◀  master
*/
```



此例中本图为无向图

04/24/18 10:53:18 D:\git-repos\data-structure-homework\07\e21.cpp

```cpp
#include <cstdio>
#include <cstring>
#include <vector>
#include <set>
#include <map>
#define MXN 1007
using namespace std;
map<set<int>, vector<int>> sts;
vector<int> v[MXN];
int n, m;
int seq[MXN], loc[MXN];
char tag[MXN];
void dfs(int root, int dep) {
    if (tag[root] == 1) {
        set<int> tmp;
        vector<int> tp;
        for (int i = loc[root]; i < dep; ++i)
            tmp.insert(seq[i]), tp.push_back(seq[i]);
        if (sts.find(tmp) != sts.cend()) return;
        sts.insert(make_pair(tmp, tp));
        return;
    }
    tag[root] = 1;
    seq[dep] = root;
    loc[root] = dep;
    for (auto i : v[root]) {
        dfs(i, dep + 1);
    }
    tag[root] = 2;
}
int main() {
    scanf("%d %d", &n, &m);
    for (int i = 0; i < m; ++i) {
        int a, b;
        scanf("%d %d", &a, &b);
        v[a].push_back(b);
    }
    for (int i = 1; i <= n; ++i)
        if (tag[i] == 0)
            dfs(i, 0);
    printf("Found %d distinct simple cycle:\n", sts.size());
    for (auto i : sts) {
        for (auto j : i.second)
            printf("%d ", j);
        putchar('\n');
    }
}
/**
 root ▶ … ≻ git-repos ≻ data-structure-homework ≻ 07 ▶ g++ e21.cpp -std=c++11
 root ▶ … ≻ git-repos ≻ data-structure-homework ≻ 07 ▶ ./a.out      ◀ master
7
11
5 6
5 7
7 6
6 2
7 4
2 4
2 3
4 1
1 2
3 1
4 3
Found 3 distinct simple cycle:
1 2 3
1 2 4 3
1 2 4
 root ▶ … ≻ git-repos ≻ data-structure-homework ≻ 07 ▶            ◀ master
*/
```



04/24/18 11:23:28 D:\git-repos\data-structure-homework\07\e22.prim.cpp

```cpp
/**--------------------
 * Prim Spanning Tree Algorithm
 *
 * Time Consumption: E \times \log{V}
 * Mem Consumption: linear
 * Author: cjsoft
 * Date: 2018/01/27
 * --------------------
 */
#include <queue>
#include <vector>
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
#define EMXN 10007
#define VMXN 107
#define E eglist
#define iterate(NODEN, _I) for (int _I = ehead[NODEN]; _I != -1; _I = eglist[_I].prev)
struct edge {
    int prev, v, w;
    edge(): prev(-1), v(0), w(0) {}
} eglist[EMXN];
int ehead[VMXN], ecur;
inline void einit() {
    ecur = 0;
    eglist[0] = edge();
    for (int i = 1; i < EMXN; ++i)
        eglist[i] = eglist[i - 1];
    for (int i = 0; i < VMXN; ++i)
        ehead[i] = -1;
}
inline void addedge(int u, int v, int w) {
    E[ecur].v = v;
    E[ecur].w = w;
    E[ecur].prev = ehead[u];
    ehead[u] = ecur++;
}
struct PII {
    int v, dis;
    PII(): v(0), dis(0) {}
    PII(int v, int dis): v(v), dis(dis) {}
    bool operator<(const PII &b) const {
        if (dis == b.dis) return v < b.v;
        return dis > b.dis;
    }
};
int G[107][107], n;
priority_queue<PII> npq;
char vis[VMXN];
int dis[VMXN];
int prim(int s) {
    int ans = 0;
    while (!npq.empty()) npq.pop();
    memset(dis, 0x3f, sizeof(dis));
    memset(vis, 0, sizeof(vis));
```

```
    dis[s] = 0;
    npq.push(PII(s, 0));
    PII tmp;
    while (!npq.empty()) {
        tmp = npq.top(), npq.pop();
        if (vis[tmp.v] || tmp.dis > dis[tmp.v]) continue;
        vis[tmp.v] = 1;
        ans += dis[tmp.v];
        for (int i = 1; i <= n; ++i) {
            if (dis[i] > G[tmp.v][i]) {
                dis[i] =  G[tmp.v][i];
                npq.push(PII(i, dis[i]));
            }
        }
    }
    return ans;
}
int main() {
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            scanf("%d", &G[i][j]);
        }
    }
    printf("%d\n", prim(1));
}
/**
 root ▶ … ≻ git-repos ≻ data-structure-homework ≻ 07 ▶ g++ e22.prim.cpp
 root ▶ … ≻ git-repos ≻ data-structure-homework ≻ 07 ▶ ./a.out      ◀  master
4
0 4 9 21
4 0 8 17
9 8 0 16
21 17 16 0
28
 root ▶ … ≻ git-repos ≻ data-structure-homework ≻ 07 ▶             ◀  master
*/
```

04/24/18 11:23:09 D:\git-repos\data-structure-homework\07\e22.kruskal.cpp

本例中通过读入邻接矩阵来建立邻接表

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#define MXN 107
using namespace std;

struct edge {
    int w, u, v;
    edge() {
        w = u = v = 0;
    }
    edge (int _a, int _b, int _c) {
        w = _a; u = _b; v = _c;
    }
    bool operator<(const edge &b) const {
        if (w < b.w) return true;
        else if (w > b.w) return false;
        return u < b.u;
    }
} egs[MXN * MXN];

int fa[MXN], cur;

inline int getfa(int n) {
    static int t, tmp;
    t = n;
    while (t != fa[t]) {
        t = fa[t];
    }
    while (n != fa[n]) {
        tmp = fa[n];
        fa[n] = t;
        n = tmp;
    }
    return t;
}

inline void init() {
    for (int i = 0; i < MXN; ++i) fa[i] = i;
}

inline void uni(int a, int b) {
    fa[getfa(b)] = getfa(a);
}
int n;
int main() {
    init();
    int t;
    cur = 0;
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            scanf("%d", &t);
            if (j > i) {
                egs[cur++] = edge(t, i, j);
            }
```

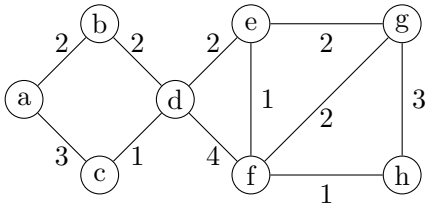
```
        }
    }
    sort(egs, egs + cur);
    t = 0;
    for (int i = 0, tt = 0; i < cur && tt < n - 1; ++i) {
        if (getfa(egs[i].u) != getfa(egs[i].v)) {
            uni(egs[i].u, egs[i].v);
            ++tt;
            t += egs[i].w;
        }
    }
    printf("%d\n", t);
    return 0;
}
/**
 root ▶ … ╱ git-repos ╱ data-structure-homework ╱ 07 ▶ g++ e22.kruskal.cpp
 root ▶ … ╱ git-repos ╱ data-structure-homework ╱ 07 ▶ ./a.out      ◀  master
4
0 4 9 21
4 0 8 17
9 8 0 16
21 17 16 0
28
 root ▶ … ╱ git-repos ╱ data-structure-homework ╱ 07 ▶             ◀  master
*/
```

求最小生成树

2017211123 褚逸豪

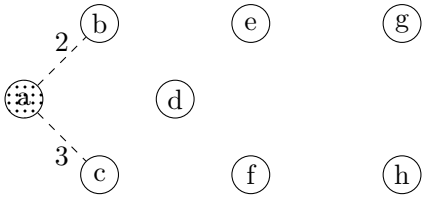
2018 年 4 月 24 日

1　初始图

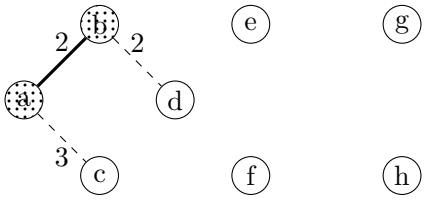


2　Prim求最小生成树

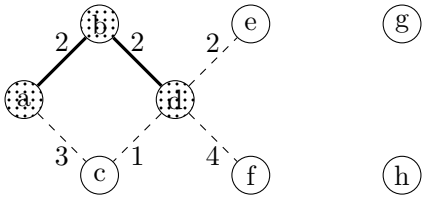
先取a为起始点



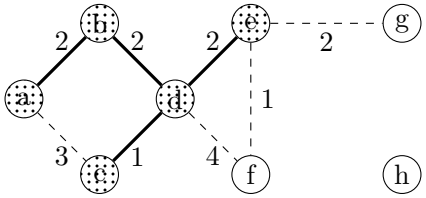
接着扩展b



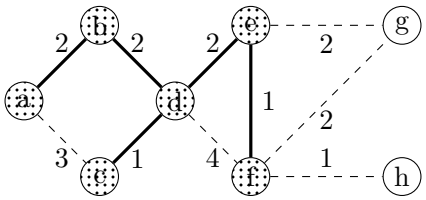
然后扩展d



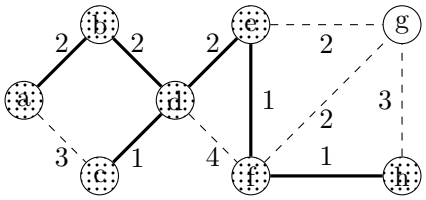
扩展c，紧接着扩展e



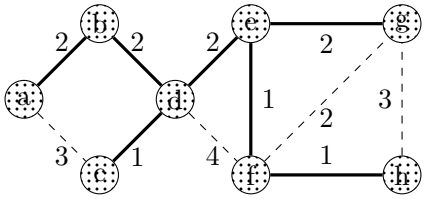
然后我们扩展f



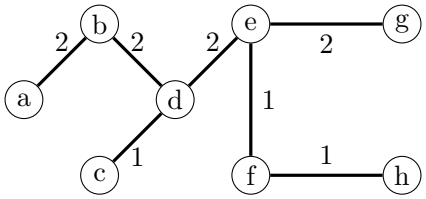
扩展h



最后吸纳g



我们得到了如下的最小生成树

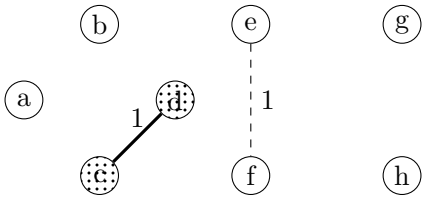


3 Kruskal求最小生成树

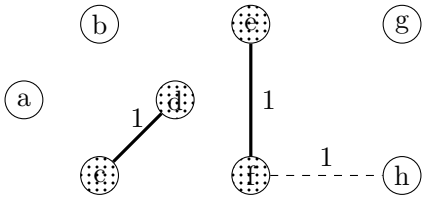
先将图中的边按权值递增序排序

1. c-d:1
2. e-f:1
3. f-h:1
4. b-d:2
5. d-e:2
6. e-g:2
7. g-f:2
8. a-b:2
9. a-c:3
10. g-h:3
11. d-f:4

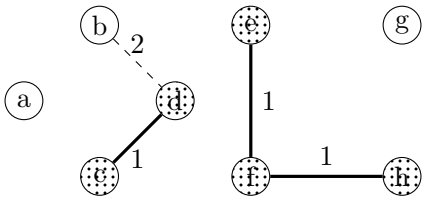
尝试并加入第一条边



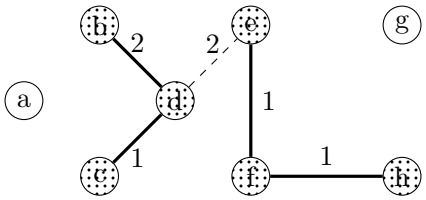
加入第二条，准备插入第三条



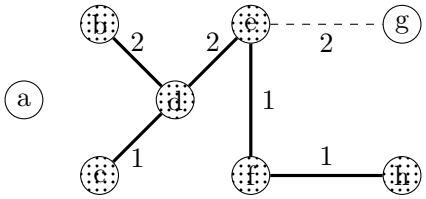
插入第三条，尝试第四条



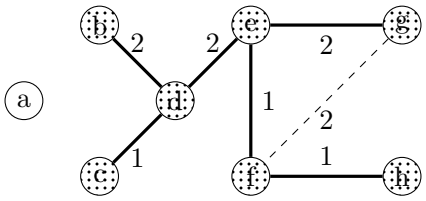
插入第四条，尝试第五条



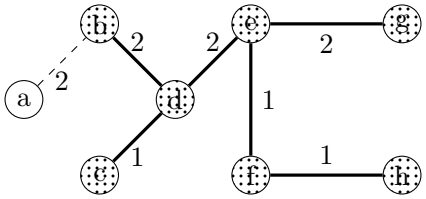
插入第五条，尝试第六条



插入第六条，在尝试第七条边的时候发现第七条边不合法



尝试第八条，合法，加入之



最后我们得到了如下最小生成树