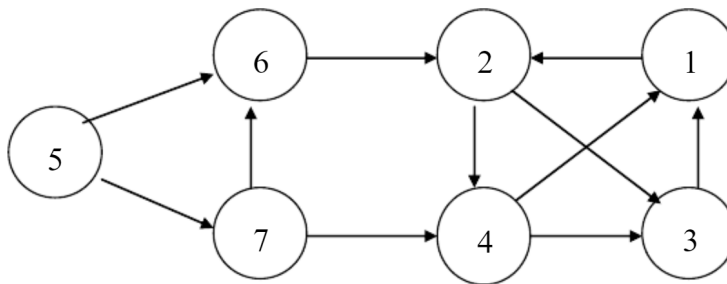


04/24/18 10:58:41 D:\git-repos\data-structure-homework\07\e20.cpp

```

1  #include <stdio>
2  #include <cstring>
3  #include <vector>
4  #define MXN 107
5  using namespace std;
6  vector<int> v[MXN];
7  bool tag[MXN];
8  int n, m;
9  int x;
10 bool dfs(int root, int dep, int target, int k) {
11     if (dep > k) return false;
12     if (target == root && dep == k) return true;
13     for (auto i : v[root]) {
14         if (!tag[i]) {
15             tag[i] = true;
16             if (dfs(i, dep + 1, target, k))
17                 return true;
18             tag[i] = false;
19         }
20     }
21     return false;
22 }
23 int main() {
24     scanf("%d %d", &n, &m);
25     for (int i = 0; i < m; ++i) {
26         int a, b;
27         scanf("%d %d", &a, &b);
28         v[a].push_back(b);
29         v[b].push_back(a);
30     }
31     int a, b, k;
32     while (~scanf("%d %d %d", &a, &b, &k)) {
33         memset(tag, 0, sizeof(tag));
34         tag[a] = true;
35         puts(dfs(a, 0, b, k) ? "YES" : "NO");
36     }
37 }
38 /**
39 root > ... > git-repos > data-structure-homework > 07 > g++ e20.cpp -std=c++11
40 root > ... > git-repos > data-structure-homework > 07 > ./a.out
41 7
42 11
43 5 6
44 5 7
45 7 6
46 6 2
47 7 4
48 2 4
49 2 3
50 4 1
51 1 2
52 3 1
53 4 3
54 1 4 5
55 YES
56 1 4 6
57 YES
58 1 4 7
59 NO
60 6 4 5
61 NO
62 6 4 3
63 YES
64 6 4 4
65 YES
66 ^Z
67 [1] + 1112 suspended ./a.out
68 root > ... > git-repos > data-structure-homework > 07 > 1 > ◀ 148 ◀ ? master
69 */

```



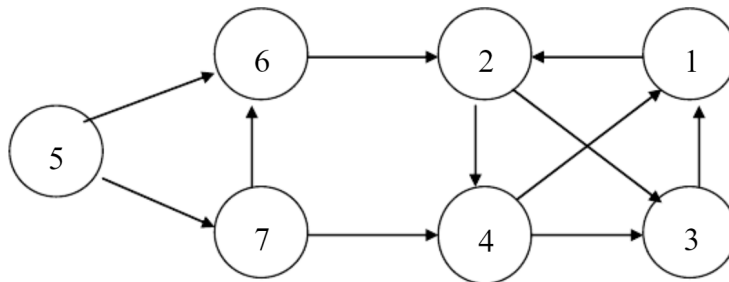
此例中本图为无向图

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

```

1  #include <stdio>
2  #include <cstring>
3  #include <vector>
4  #include <set>
5  #include <map>
6  #define MXN 1007
7  using namespace std;
8  map<set<int>, vector<int>> sts;
9  vector<int> v[MXN];
10 int n, m;
11 int seq[MXN], loc[MXN];
12 char tag[MXN];
13 void dfs(int root, int dep) {
14     if (tag[root] == 1) {
15         set<int> tmp;
16         vector<int> tp;
17         for (int i = loc[root]; i < dep; ++i)
18             tmp.insert(seq[i]), tp.push_back(seq[i]);
19         if (sts.find(tmp) != sts.cend()) return;
20         sts.insert(make_pair(tmp, tp));
21         return;
22     }
23     tag[root] = 1;
24     seq[dep] = root;
25     loc[root] = dep;
26     for (auto i : v[root]) {
27         dfs(i, dep + 1);
28     }
29     tag[root] = 2;
30 }
31 int main() {
32     scanf("%d %d", &n, &m);
33     for (int i = 0; i < m; ++i) {
34         int a, b;
35         scanf("%d %d", &a, &b);
36         v[a].push_back(b);
37     }
38     for (int i = 1; i <= n; ++i)
39         if (tag[i] == 0)
40             dfs(i, 0);
41     printf("Found %d distinct simple cycle:\n", sts.size());
42     for (auto i : sts) {
43         for (auto j : i.second)
44             printf("%d ", j);
45         putchar('\n');
46     }
47 }
48 /**
49 root ► ... > git-repos > data-structure-homework > 07 ► g++ e21.cpp -std=c++11
50 root ► ... > git-repos > data-structure-homework > 07 ► ./a.out ◀ master
51 7
52 11
53 5 6
54 5 7
55 7 6
56 6 2
57 7 4
58 2 4
59 2 3
60 4 1
61 1 2
62 3 1
63 4 3
64 Found 3 distinct simple cycle:
65 1 2 3
66 1 2 4 3
67 1 2 4
68 root ► ... > git-repos > data-structure-homework > 07 ► ◀ master
69 */

```



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

```

1  /**-----
2   * Prim Spanning Tree Algorithm
3   *
4   * Time Consumption: E \times \log{V}
5   * Mem Consumption: linear
6   * Author: cjsoft
7   * Date: 2018/01/27
8   * -----
9   */
10 #include <queue>
11 #include <vector>
12 #include <iostream>
13 #include <cstdio>
14 #include <cstring>
15 using namespace std;
16 #define EMXN 10007
17 #define VMXN 107
18 #define E eglist
19 #define iterate(NODEN, _I) for (int _I = ehead[NODEN]; _I != -1; _I =
    eglist[_I].prev)
20 struct edge {
21     int prev, v, w;
22     edge(): prev(-1), v(0), w(0) {}
23 } eglist[EMXN];
24 int ehead[VMXN], ecur;
25 inline void einit() {
26     ecur = 0;
27     eglist[0] = edge();
28     for (int i = 1; i < EMXN; ++i)
29         eglist[i] = eglist[i - 1];
30     for (int i = 0; i < VMXN; ++i)
31         ehead[i] = -1;
32 }
33 inline void addedge(int u, int v, int w) {
34     E[ecur].v = v;
35     E[ecur].w = w;
36     E[ecur].prev = ehead[u];
37     ehead[u] = ecur++;
38 }
39 struct PII {
40     int v, dis;
41     PII(): v(0), dis(0) {}
42     PII(int v, int dis): v(v), dis(dis) {}
43     bool operator<(const PII &b) const {
44         if (dis == b.dis) return v < b.v;
45         return dis > b.dis;
46     }
47 };
48 int G[107][107], n;
49 priority_queue<PII> npq;
50 char vis[VMXN];
51 int dis[VMXN];
52 int prim(int s) {
53     int ans = 0;
54     while (!npq.empty()) npq.pop();
55     memset(dis, 0x3f, sizeof(dis));
56     memset(vis, 0, sizeof(vis));

```

```

57     dis[s] = 0;
58     npq.push(PII(s, 0));
59     PII tmp;
60     while (!npq.empty()) {
61         tmp = npq.top(), npq.pop();
62         if (vis[tmp.v] || tmp.dis > dis[tmp.v]) continue;
63         vis[tmp.v] = 1;
64         ans += dis[tmp.v];
65         for (int i = 1; i <= n; ++i) {
66             if (dis[i] > G[tmp.v][i]) {
67                 dis[i] = G[tmp.v][i];
68                 npq.push(PII(i, dis[i]));
69             }
70         }
71     }
72     return ans;
73 }
74 int main() {
75     scanf("%d", &n);
76     for (int i = 1; i <= n; ++i) {
77         for (int j = 1; j <= n; ++j) {
78             scanf("%d", &G[i][j]);
79         }
80     }
81     printf("%d\n", prim(1));
82 }
83 /**
84 root ► ... > git-repos > data-structure-homework > 07 ► g++ e22.prim.cpp
85 root ► ... > git-repos > data-structure-homework > 07 ► ./a.out ◀ master
86 4
87 0 4 9 21
88 4 0 8 17
89 9 8 0 16
90 21 17 16 0
91 28
92 root ► ... > git-repos > data-structure-homework > 07 ► ◀ master
93 */

```

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

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  #define MXN 107
6  using namespace std;
7
8  struct edge {
9      int w, u, v;
10     edge() {
11         w = u = v = 0;
12     }
13     edge (int _a, int _b, int _c) {
14         w = _a; u = _b; v = _c;
15     }
16     bool operator<(const edge &b) const {
17         if (w < b.w) return true;
18         else if (w > b.w) return false;
19         return u < b.u;
20     }
21 } egs[MXN * MXN];
22
23 int fa[MXN], cur;
24
25 inline int getfa(int n) {
26     static int t, tmp;
27     t = n;
28     while (t != fa[t]) {
29         t = fa[t];
30     }
31     while (n != fa[n]) {
32         tmp = fa[n];
33         fa[n] = t;
34         n = tmp;
35     }
36     return t;
37 }
38
39 inline void init() {
40     for (int i = 0; i < MXN; ++i) fa[i] = i;
41 }
42
43 inline void uni(int a, int b) {
44     fa[getfa(b)] = getfa(a);
45 }
46
47 int n;
48 int main() {
49     init();
50     int t;
51     cur = 0;
52     scanf("%d", &n);
53     for (int i = 1; i <= n; ++i) {
54         for (int j = 1; j <= n; ++j) {
55             scanf("%d", &t);
56             if (j > i) {
57                 egs[cur++] = edge(t, i, j);

```

```

58     }
59 }
60 sort(egs, egs + cur);
61 t = 0;
62 for (int i = 0, tt = 0; i < cur && tt < n - 1; ++i) {
63     if (getfa(egs[i].u) != getfa(egs[i].v)) {
64         uni(egs[i].u, egs[i].v);
65         ++tt;
66         t += egs[i].w;
67     }
68 }
69 printf("%d\n", t);
70 return 0;
71 }
72 /**
73 root ► ... > git-repos > data-structure-homework > 07 ► g++ e22.kruskal.cpp
74 root ► ... > git-repos > data-structure-homework > 07 ► ./a.out ◀ ? master
75 4
76 0 4 9 21
77 4 0 8 17
78 9 8 0 16
79 21 17 16 0
80 28
81 root ► ... > git-repos > data-structure-homework > 07 ► ◀ ? master
82 */

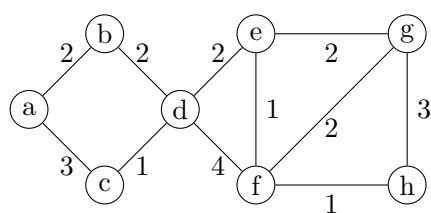
```

# 求最小生成树

2017211123 褚逸豪

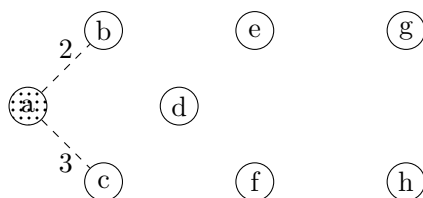
2018 年 4 月 24 日

## 1 初始图

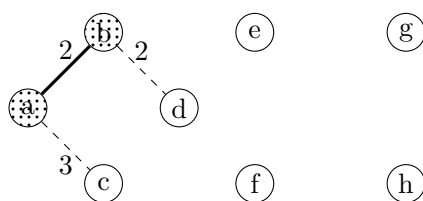


## 2 Prim求最小生成树

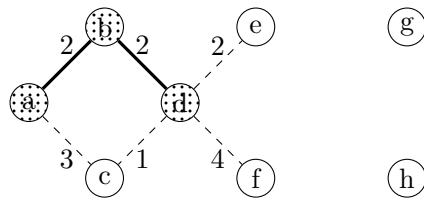
先取 $a$ 为起始点



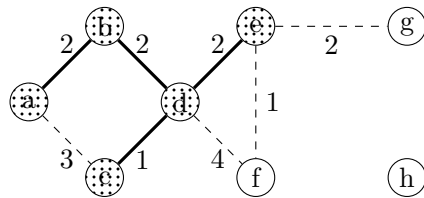
接着扩展 $b$



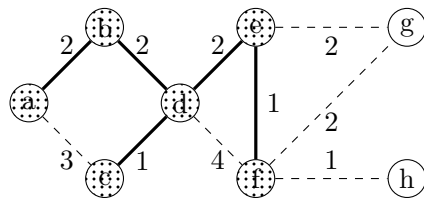
然后扩展 $d$



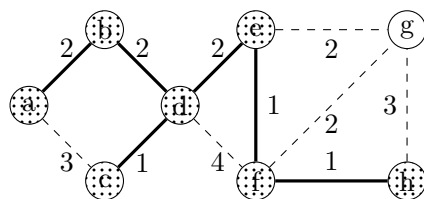
扩展 $c$ , 紧接着扩展 $e$



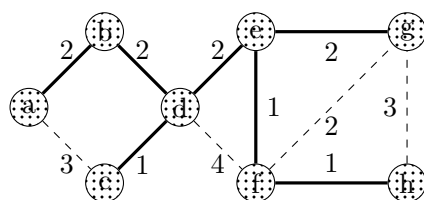
然后我们扩展 $f$



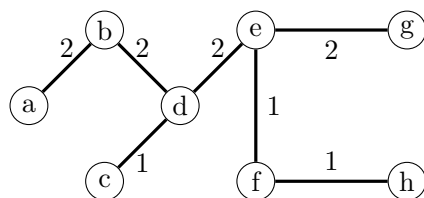
扩展 $h$



最后吸纳 $g$



我们得到了如下的最小生成树



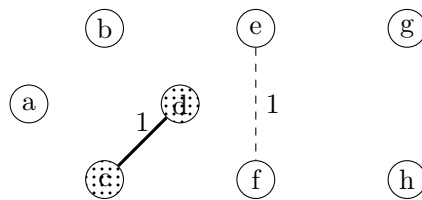


### 3 Kruskal求最小生成树

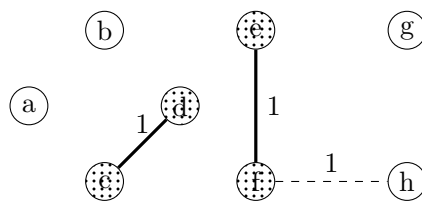
先将图中的边按权值递增序排序

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

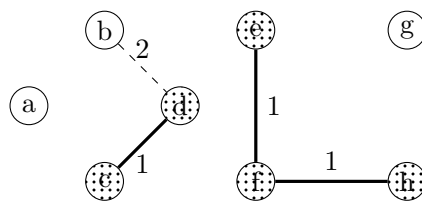
尝试并加入第一条边



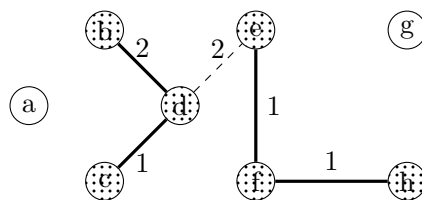
加入第二条，准备插入第三条



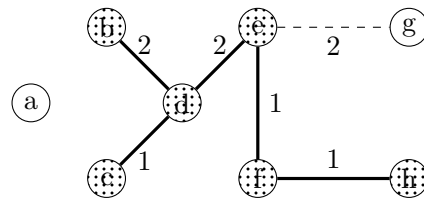
插入第三条，尝试第四条



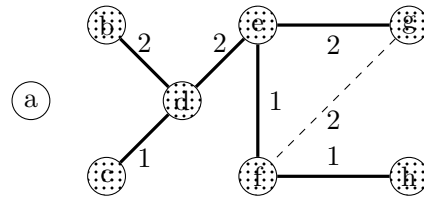
插入第四条，尝试第五条



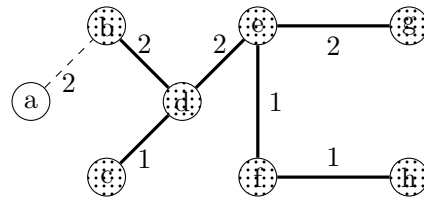
插入第五条，尝试第六条



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

