省选基础算法

雷宇辰

2017 年 2 月 13 日

目录

1	day1 图论													1
	1.1 有向图强连通分量的 Tarja	1 算法		•										1
	1.2 图的割点、桥与双连通分量													4

1 day1 图论

1.1 有向图强连通分量的 Tarjan 算法

定义 在有向图 G 中,如果两个顶点 u,v 间存在一条路径 u 到 v 的路径且也存在一条 v 到 u 的路径,则称这两个顶点 u,v 是强连通的 (strongly connected)。如果有向图 G 的每两个顶点都强连通,称 G 是一个强连通图。有向非强连通图的极大强连通子图,称为强连通分量 (strongly connected components)。若将有向图中的强连通分量都缩为一个点,则原图会形成一个 DAG(有向无环图)。

极大强连通子图 G 是一个极大强连通子图当且仅当 G 是一个强连通子图且不存在另一个强连通子图 G'使得 G 是 G'的真子集。

```
Tarjan 算法 定义 dfn(u) 为结点 u 搜索的次序编号,给出函数 low(u) 使得 low(u) = min {  dfn(u), \\ low(v), \quad (u,v) 为树枝边,u 为 v 的父结点  dfn(v) \qquad (u,v) 为后向边或指向栈中结点的横叉边
```

当结点 u 的搜索过程结束后,若 df n(u) = low(u),则以 u 为根的搜索子树上所有还在栈中的结点是一个强连通分量。

代码

tarjan - SCC

```
void tarjan(int u)
 1
 2
 3
        dfn[u] = low[u] = ++idx;
 4
        st[top++] = u;
 5
        for (Edge cur : G[u])
             if (!dfn[cur.to])
 6
 7
                 tarjan(cur.to),
8
                 low[u] = min(low[u], low[cur.to]);
9
             else if (!scc[cur.to])
10
                 low[u] = min(low[u], dfn[cur.to]);
11
        if (dfn[u] == low[u] \&\& ++cnt)
12
             do scc[st[--top]] = cnt;
13
             while (st[top] != u);
14
```

练习题

POJ2186/BZOJ1051 - Popular Cows 双倍的快乐

Popular Cows

```
1  #include <cstdio>
2  inline int min(int a, int b) { return a < b ? a : b; }
3  int head[10010], next[50010], to[50010], ecnt;
4  int dfn[10010], low[10010], stk[10010], scc[10010], top, idx, scccnt;
5  bool instk[10010];</pre>
```

```
int deg[10010];
    inline void addEdge(int f, int t)
8
9
         ecnt++;
         next[ecnt] = head[f];
10
11
         head[f] = ecnt;
12
         to[ecnt] = t;
13
    }
    void tarjan(int x)
14
15
         dfn[x] = low[x] = ++idx;
16
17
         instk[stk[top++] = x] = true;
18
         for (int cur = head[x]; cur; cur = next[cur])
19
             if (!dfn[to[cur]])
20
                 tarjan(to[cur]), low[x] = min(low[x], low[to[cur]]);
             else if (instk[to[cur]])
21
                 low[x] = min(low[x], dfn[to[cur]]);
22
23
         if (dfn[x] == low[x])
24
         {
25
             scccnt++;
26
             do
27
28
                 top---;
29
                 scc[stk[top]] = scccnt;
30
                 instk[stk[top]] = false;
             } while (stk[top] != x);
31
32
         }
33
34
    int main()
35
36
         int n, m;
37
         scanf("%d%d", &n, &m);
         for (int i = 0, x, y; i < m; i++)
38
39
             scanf("%d%d", &x, &y);
40
41
             addEdge(x, y);
42
43
         for (int i = 1; i <= n; i++)
             if (!dfn[i])
44
45
                 tarjan(i);
         for (int i = 1; i \leftarrow n; i++)
46
47
             for (int cur = head[i]; cur; cur = next[cur])
48
                 if (scc[i] != scc[to[cur]])
49
                     deg[scc[i]]++;
         int zcnt = 0, id = 0;
50
         for (int i = 1; i \leftarrow scccnt; i++)
51
52
             if (deg[i] == 0)
53
                 zcnt++, id = i;
54
         if (zcnt != 1)
55
             putchar('0');
         else
56
57
58
             int ans = 0;
59
             for (int i = 1; i <= n; i++)
60
                 if (scc[i] == id)
61
                      ans++;
62
             printf("%d", ans);
63
         }
64
         return 0;
65
```

POJ3180 - The Cow Prom The N (2 <= N <= 10,000) cows are so excited.

The Cow Prom

```
#include <cstdio>
    inline int min(int a, int b) { return a < b ? a : b; }</pre>
    const int maxn = 100010;
 3
    int head[maxn], next[maxn << 1], to[maxn << 1], ecnt, n, m;</pre>
 4
 5
    int dfn[maxn], scc[maxn], cnt[maxn], scccnt, stk[maxn], low[maxn], idx, top;
    inline void addEdge(int f, int t)
6
 7
8
        ecnt++;
9
        next[ecnt] = head[f];
10
        head[f] = ecnt;
11
        to[ecnt] = t;
12
    }
13
    void tarjan(int x)
14
15
        dfn[x] = low[x] = ++idx;
16
         stk[top++] = x;
17
        for (int i = head[x]; i; i = next[i])
18
             if (!dfn[to[i]])
19
                 tarjan(to[i]), low[x] = min(low[x], low[to[i]]);
20
             else if (!scc[to[i]])
21
                 low[x] = min(low[x], dfn[to[i]]);
22
        if (dfn[x] == low[x])
23
        {
24
             scccnt++;
25
                 scc[stk[--top]] = scccnt;
26
27
             while (stk[top] != x);
28
        }
29
30
    int main()
31
         scanf("%d%d", &n, &m);
32
        for (int i = 0, x, y; i < m; i++)
33
34
35
             scanf("%d%d", &x, &y);
36
             addEdge(x, y);
37
        }
        for (int i = 1; i <= n; i++)
38
39
             if (!dfn[i]) tarjan(i);
40
         int ans = 0;
41
         for (int i = 1; i <= n; i++) cnt[scc[i]]++;
        for (int i = 1; i <= scccnt; i++)
42
43
             if (cnt[i] > 1) ans++;
        printf("%d", ans);
44
        return 0;
45
46
```

POJ1236 - Network of Schools 强连通分量缩点求出度为 0 的和入度为 0 的分量个数

Network of Schools

```
#include <cstdio>
inline int min(int a, int b) { return a < b ? a : b; }

const int maxn = 110, maxm = 10100;

int head[maxn], next[maxm], to[maxm], ecnt, f[maxn], g[maxn];

inline void addEdge(int f, int t)

{
    ecnt++;</pre>
```

```
8
        next[ecnt] = head[f];
 9
        head[f] = ecnt;
10
        to[ecnt] = t;
11
12
    int dfn[maxn], low[maxn], stk[maxn], scc[maxn], scccnt, top, idx;
13
    void tarjan(int x)
14
15
        dfn[x] = low[x] = ++idx;
        stk[top++] = x;
16
17
        for (int i = head[x]; i; i = next[i])
             if (!dfn[to[i]])
18
19
                 tarjan(to[i]), low[x] = min(low[x], low[to[i]]);
20
             else if (!scc[to[i]])
21
                 low[x] = min(low[x], dfn[to[i]]);
22
        if (dfn[x] == low[x])
23
24
             scccnt++;
25
26
                 scc[stk[--top]] = scccnt;
27
             while (stk[top] != x);
28
        }
29
30
    int main()
31
    {
32
        int n;
        scanf("%d", &n);
33
        for (int i = 1, x; i <= n; i++)
34
             for (scanf("%d", &x); x; scanf("%d", &x))
35
36
                 addEdge(i, x);
37
        for (int i = 1; i <= n; i++)
38
             if (!dfn[i]) tarjan(i);
39
        for (int i = 1; i <= n; i++)
             for (int j = head[i]; j; j = next[j])
40
                 if (scc[i] != scc[to[j]])
41
42
                     f[scc[i]]++, g[scc[to[j]]]++;
43
        int ans1 = 0, ans2 = 0;
44
        if (scccnt == 1)
45
             printf("1\n0");
46
        else
47
        {
             for (int i = 1; i <= scccnt; i++)
48
49
                 ans1 += f[i] == 0, ans2 += g[i] == 0;
             printf("%d\n%d", ans2, ans1 > ans2 ? ans1 : ans2);
50
51
        }
        return 0;
52
53
    }
```

1.2 图的割点、桥与双连通分量

定义

点连通度与边连通度 在一个**无向连通图**中,如果有一个顶点集合 V,删除顶点集合 V,以及与 V中顶点相连(至少有一端在 V 中)的所有边后,原图**不连通**,就称这个点集 V 为**割点集合**。一个图的**点连通度**的定义为:最小割点集合中的顶点数。

类似的,如果有一个边集合,删除这个边集合以后,原图不连通,就称这个点集为割边集合。

双连通图、割点与桥 如果一个无向连通图的点连通度大于 1,则称该图是点双连通的 (point biconnected),简称双连通或重连通。一个图有割点,当且仅当这个图的点连通度为 1,则割点集合的唯一元素被称为割点 (cut point),又叫关节点 (articulation point)。一个图可能有多个割点。

如果一个无向连通图的**边连通度大于** 1,则称该图是**边双连通的 (edge biconnected)**,简称双连通或 重连通。一个图有桥,当且仅当这个图的边连通度为 1,则割边集合的唯一元素被称为桥 (bridge),又 叫关节边 (articulation edge)。一个图可能有多个桥。

可以看出,点双连通与边双连通都可以简称为双连通,它们之间是有着某种联系的,下文中提到的双连通,均既可指点双连通,又可指边双连通。(但这并不意味着它们等价)

双连通分量(分支): 在图 G 的所有子图 G'中,如果 G'是双连通的,则称 G'为双连通子图。如果一个双连通子图 G'它不是任何一个双连通子图的真子集,则 G'为极大双连通子图。双连通分量 (biconnected component),或重连通分量,就是图的极大双连通子图。特殊的,点双连通分量又叫做块。

```
Tarjan 算法 给出函数 low(u) 使得
```

```
low(u) = min {  dfn(u), \\ low(v), \quad (u,v) 为树枝边 (父子边)  dfn(v) \quad (u,v) 为后向边 (返祖边) 等价于 dfn(v) < dfn(u) 且 v 不为 u 的父亲结点 }
```

代码

tarjan - BCC

```
1
   void tarjan(int u, int p)
2
   {
3
       dfn[u] = low[u] = ++idx;
4
       for (int e = head[u]; e; e = next[e])
5
           if (!dfn[to[e]])
               tarjan(to[e], u), low[u] = min(low[u], low[to[e]]);
6
7
           else if (to[e] != p)
8
               low[u] = min(low[u], dfn[to[e]]);
```

练习题

POJ3177 - Redundant Paths 将一张有桥图通过加边变成边双连通图,至少要加 $\frac{leaf+1}{2}$ 条边。

Redundant Paths

```
1  #include <cstdio>
2  inline int min(int a, int b) { return a < b ? a : b; }
3  int head[5010], to[20010], next[20010], ecnt, map[5010][5010];
4  int dfn[5010], low[5010], idx, cnt[5010];
5  void addEdge(int f, int t)
6  {
7     ecnt++;
8     next[ecnt] = head[f];
9  head[f] = ecnt;</pre>
```

```
10
         to[ecnt] = t;
11
    }
12
    void tarjan(int u, int p)
13
14
         dfn[u] = low[u] = ++idx;
15
         for (int e = head[u]; e; e = next[e])
16
             if (!dfn[to[e]])
17
                 tarjan(to[e], u), low[u] = min(low[u], low[to[e]]);
             else if (to[e] != p)
18
19
                 low[u] = min(low[u], dfn[to[e]]);
20
21
    int main()
22
    {
23
         int n, m;
24
         scanf("%d%d", &n, &m);
25
         for (int i = 1, x, y; i \leftarrow m; i++)
26
             scanf("%d%d", &x, &y);
27
28
             if (!map[x][y])
29
                 addEdge(x, y);
30
31
                 addEdge(y, x);
32
                 map[x][y] = map[y][x] = true;
33
34
         }
35
         tarjan(1, 0);
         for (int i = 1; i <= n; i++)
36
37
             for (int e = head[i]; e; e = next[e])
38
                 if (low[to[e]] != low[i])
39
                      cnt[low[i]]++;
40
         int ans = 0;
41
         for (int i = 1; i <= n; i++)
42
             ans += cnt[i] == 1;
         printf("%d", (ans + 1) >> 1);
43
44
         return 0;
45
```

POJ1523 - SPF 求割点与删除这个点之后有多少个连通分量

Redundant Paths

```
#include <cstdio>
 1
 2
    #include <cctype>
    #include <cstring>
    #define clz(X) memset(X, 0, sizeof(X))
 5
    inline int max(int a, int b) { return a > b ? a : b; }
    inline int min(int a, int b) { return a < b ? a : b; }</pre>
 6
 7
    inline void read(int &x)
8
9
         int ch = x = 0;
10
         while (!isdigit(ch = getchar()));
         for (; isdigit(ch); ch = getchar())
11
             x = x * 10 + ch - '0';
12
13
14
    int map[1010][1010], range;
15
    int dfn[1010], low[1010], idx;
    int son, subnet[1010];
16
17
    void tarjan(int u)
18
19
         dfn[u] = low[u] = ++idx;
20
         for (int v = 1; v \leftarrow range; v++)
21
             if (map[u][v])
```

```
if (!dfn[v])
22
23
24
                     tarjan(v);
                     low[u] = min(low[u], low[v]);
25
26
                     if (low[v] >= dfn[u])
27
                         (u == 1 ? son : subnet[u])++;
28
                 }
29
                 else
30
                     low[u] = min(low[u], dfn[v]);
31
32
    int main()
33
34
         int x, y, T = 0;
35
         while (read(x), x)
36
37
             clz(map), clz(dfn), clz(low), clz(subnet), son = idx = 0;
38
             read(y);
39
             map[x][y] = map[y][x] = 1;
40
             range = max(x, y);
41
             while (read(x), x)
42
43
                 read(y);
44
                 map[x][y] = map[y][x] = 1;
45
                 range = max(range, max(x, y));
46
47
            printf("Network #%d\n", ++T);
48
             tarjan(1);
             bool flag = false;
49
             if (son > 1) subnet[1] = son - 1;
50
51
             for (int i = 1; i <= range; i++)
52
                 if (subnet[i])
53
                     printf(" SPF node %d leaves %d subnets \n", i, subnet[i] + 1),\\
                     flag = true;
54
             if (!flag)
55
                 puts(" No SPF nodes");
56
57
             putchar('\n');
58
59
         return 0;
60
    }
```