实验舱
青少年编程
走近科学 走进名校

提高算法班

无向图的连通性

Mas

割点、桥

对于连通图 $G = (V, E)$

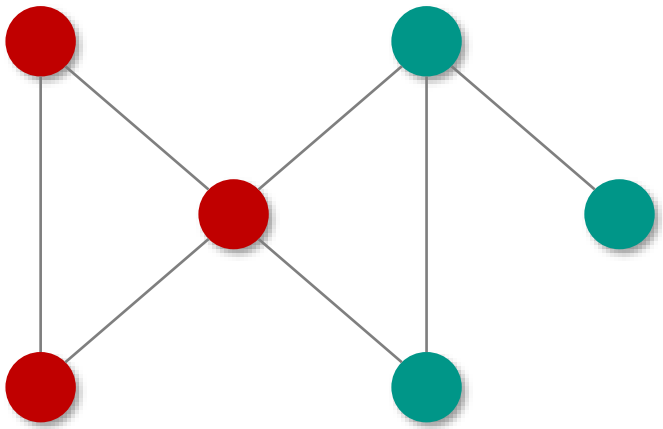
若 $V' \subseteq V$ 且 $G[V \setminus V']$(即从 G 中删去 V' 中的点)不是连通图,则 V' 是图 G 的一个 点割集 (Vertex cut/Separating set)

大小为一的点割集又被称作 **割点** (Cut vertex)

对于连通图 $G = (V, E)$

若 $E' \subseteq E$ 且 $G[E \setminus E']$(即从 G 中删去 E' 中的边)不是连通图,则 E' 是图 G 的一个 边割集 (Edge cut)

大小为一的边割集又被称作 **桥** (Bridge)



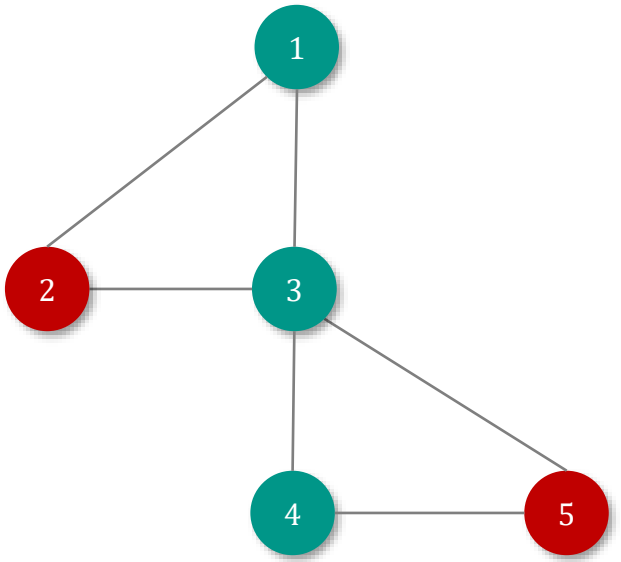
割点

low_u 定义为能够回溯到的最早的已经在栈中的结点

设以 u为根的子树为 subtree_u

low_u 定义为以下结点的 dfn 的最小值:

subtree_u 中的结点、从 subtree_u 通过**一条**不在搜索树上的边能到达的结点



无向图不存在横叉边,前向边则对 low_u 没有影响

若 u 存在一个子结点 v 满足 $\text{low}_v \geq \text{dfn}_u$, 说明 v 无法通过它的子树到达比 u 的 DFS 序更小的节点

走子树走不通，v 若想到达这样的点，只能选择经过它的父节点 u

即删去u，那么 v 和 DFS 序小于 u 点的点就分开了

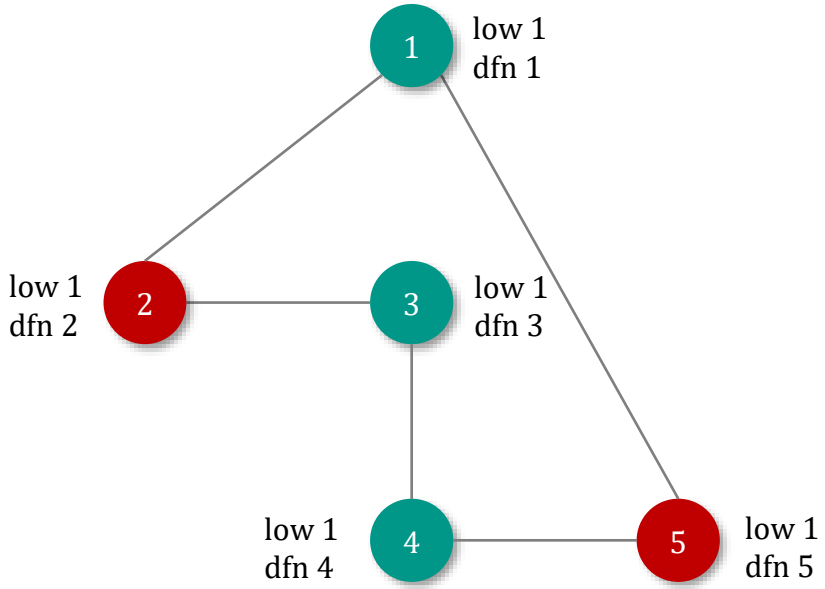
割点

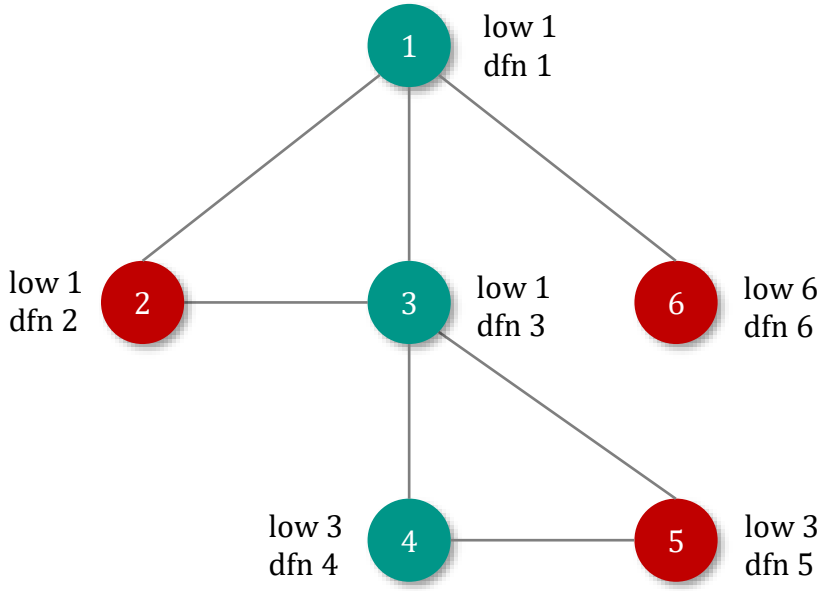
如果发现上述情况一般可以说 u 是割点了，但有一种特殊情况例外：u 是 DFS 生成树的根节点

此时整个连通分量都不存在比 u 的 DFS 序更小的点

对于根节点它如果有两个以上子节点,那么它就是割点

显然删除根节点后这两个分支将会互不相连





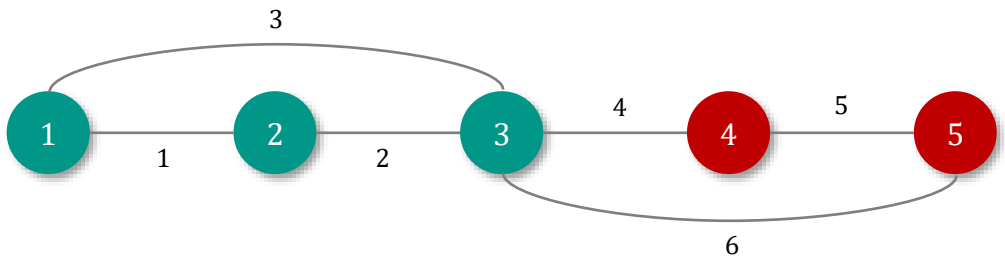
割点

```
void tarjan(int u, int root)
{
  dfn[u] = low[u] = ++idx;
  int cnt = 0;
  for (int i = head[u]; i; i = e[i].next)
  {
    int v = e[i].v;
    if (!dfn[v])
    {
      tarjan(v, root);
      low[u] = min(low[u], low[v]);
      if (low[v] >= dfn[u] && (u != root || ++cnt > 1))
        isCut[u] = true;
    }
    else
      low[u] = min(low[u], dfn[v]);
  }
}
```

时间复杂度?

是否需要排除无向图的相反边?

能否只使用 low_v 更新 low_u



#899、割点(割顶)

题目描述

给出一个 n 个点, m 条边的无向图,求图的割点

输入格式

第一行输入两个正整数 n, m

下面 m 行每行输入两个正整数 x, y 表示 x 到 y 有一条边

输出格式

第一行输出割点个数

第二行按照节点编号从小到大输出节点,用空格隔开

数据规模

对于全部数据, $1 \leq n \leq 2 \times 10^4, 1 \leq m \leq 1 \times 10^5$

点的编号均大于 0 小于等于 n 。

输入样例

```
6 7
1 2
1 3
1 4
2 5
3 5
4 5
5 6
```

输出样例

```
1
5
```

#698、电力

题目描述

求一个图删除一个点之后,联通块最多有多少

输入格式

多组数据

第一行两个整数 P, C 表示点数和边数

接下来 C 行每行两个整数 p_1, p_2 ,表示 p_1 与 p_2 有边连接,保证无重边

读入以 `0 0` 结束。

输出格式

输出若干行,表示每组数据的结果

数据范围与提示

对于全部的数据 $1 \leq P \leq 10000, C \geq 0, 0 \leq p_1, p_2 < P$

样例输入

```
3 3
0 1
0 2
2 1
4 2
0 1
2 3
3 1
1 0
0 0
```

样例输出

```
1
2
2
```

#698、电力

记原图连通块个数为 cnt

若删除的点不是割点,那么删除并不会增加连通块

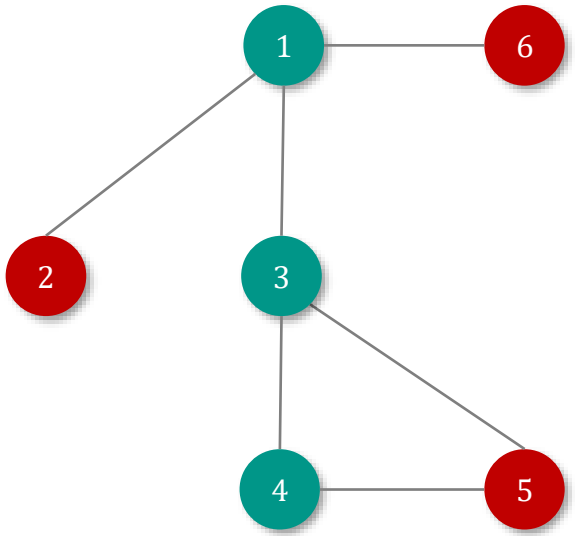
如果是割点

DFS 生成树根节点,变为分支数量个连通块

非根节点,变为分支数量个 + 1 个连通块

在 Tarjan 中求割点并统计, 删点能得多少个连通块记为 maxCnt

最终答案为 cnt + maxCnt − 1



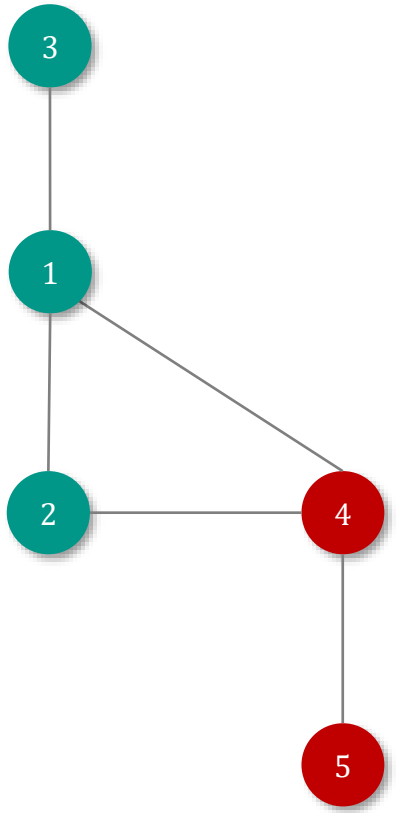
桥

low_u 定义为能够回溯到的最早的已经在栈中的结点

设以 u为根的子树为 subtree_u

low_u 定义为以下结点的 dfn 的最小值:

subtree_u 中的结点、从 subtree_u 通过**一条**不在搜索树上的边能到达的结点



如果 u 是 v 的父节点, 并且 $\text{low}_v > \text{dfn}_u$, 那么 $u \leftrightarrow v$ 是桥

证明

若 $u \leftrightarrow v$ 不是桥,那么删掉这条边后 v 一定有其他路径可以到达 u

无向图没有横叉边 , 想要到达 u 只能通过子树走反向边实现 , 那么 $\text{low}_v \leq \text{dfn}_u$ 应该成立

这与条件矛盾 , 因此 $u \leftrightarrow v$ 桥

桥

```cpp
int n, m, head[20005], dfn[20001], low[20001], pos = 1, u, v, idx;
void addEdge(int u, int v)
{
  e[++pos] = {u, v, head[u]};
  head[u] = pos;
}
vector<Edge> ans;
void tarjan(int u, int eIdx)
{
  dfn[u] = low[u] = ++idx;
  int cnt = 0;
  for (int i = head[u]; i; i = e[i].next)
  {
    int v = e[i].v;
    if (!dfn[v])
    {
      tarjan(v, i);
      low[u] = min(low[u], low[v]);
      if (low[v] > dfn[u])
        ans.push_back({u, v});
    }
    else if (i != (eIdx ^ 1))
      low[u] = min(low[u], dfn[v]);
  }
}
```

时间复杂度?

为什么边集数组下标从 2 开始?

能否只使用 low_v 更新 low_u

能否通过加一个参数 fa 来排除 无向图的相反边?

#1271、割边(桥)

题目描述

A 国派出将军 Mas，对 B 国进行战略性措施，以解救涂炭的生灵。

B 国有 n 个城市，这些城市以铁路相连。任意两个城市都可以通过铁路直接或者间接到达。 Mas 发现有些铁路被毁坏之后，某两个城市无法互相通过铁路到达。这样的铁路就被称为 $key\ road$ 。

Mas 为了尽快使该国的物流系统瘫痪，希望炸毁铁路，以达到存在某两个城市无法互相通过铁路到达的效果。 然而，只有一发炮弹(A 国国会不给钱了)。所以，他能轰炸哪一条铁路呢?

输入格式

第一行 n, m，分别表示有 n 个城市，总共 m 条铁路。

以下 m 行,每行两个整数 a, b，表示城市 a 和城市 b 之间有铁路直接连接。

输出格式

输出有若干行。 每行包含两个数字 a, b，其中 $a < b$，表示 a, b 是 $key\ road$ 。

请注意：输出时，所有的数对 a, b 必须按照 a 从小到大排序输出；如果 a 相同，则根据 b 从小到大排序。

数据规模

对于全部数据， $1 \leq n \leq 2 \times 10^4$ ， $1 \leq m \leq 1 \times 10^5$ 。点的编号均大于 0 小于等于 n 。

双连通分量 DCC

在一张连通的无向图中

对于两个点 u 和 v, 若无论删去哪条边(只能删去一条)都不能使它们不连通,那么 u 和 v **边双连通 (** $2-\text{edge}-\text{connected}$ **)**

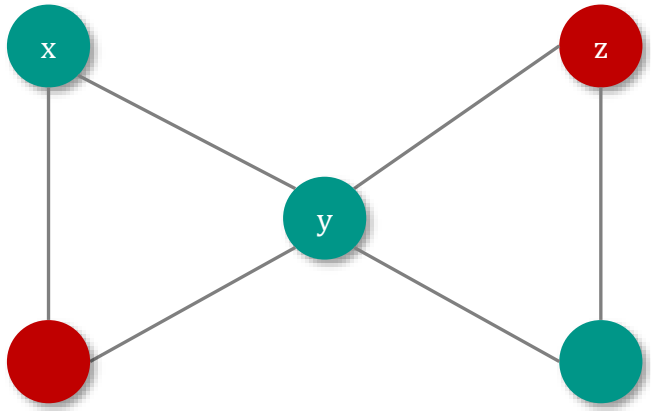
对于两个点 u 和 v, 若无论删去哪个点(只能删一个且不能删 u 或 v)都不能使它们不连通, 那么 u 和 v **点双连通(** Biconnected **)**

边双连通具有传递性

若 x,y 边双连通, y,z 边双连通, 则 x,z 边双连通

点双连通 不 具有传递性

如图, x,y 点双连通, y,z 点双连通, 而 x,z **不** 点双连通



与连通分量类似有

点双连通分量(Biconnected component)(极大点双连通子图)

边双连通分量 ($2-\text{edge}-\text{connected}$ component)(极大边双连通子图)

点双连通分量 v-DCC

若一张无向连通图不存在割点, 则称它为点双连通图

无向图的极大点双连通子图被称为 **点双连通分量** (v − DCC)

一张无向连通图是点双连通分量, 当且仅当满足下列两个条件之一:

- 图的顶点数不超过 2
- 图中任意两点都同时包含在至少一个简单环

若某个节点为孤立点, 则它自己单独构成一个 v − DCC

除了孤立点外点双连通分量的大小至少为 2

根据 v − DCC 定义中的"极大"性, 原图割点可能属于多个 v − DCC

点双连通分量 v-DCC

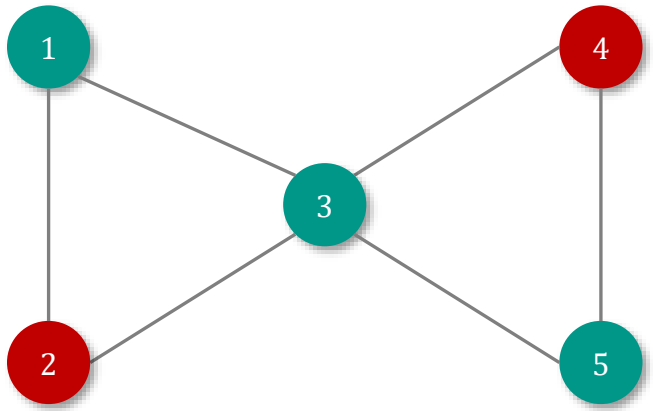
在 Tarjan 算法过程中维护一个栈

按照如下方法维护栈中的元素：

1. 当一个节点第一次被访问时，该节点入栈
2. 当割点判定方法则中的条件 $\mathrm{dfn}_u \le \mathrm{low}_v$ 成立时，无论 u 是否为根都要：

   从栈顶不断弹出节点，直至节点 v 被弹出

   刚才弹出的所有节点与节点 u 一起构成一个 v－DCC




```
void tarjan(int u, int root)
{
  dfn[u] = low[u] = ++idx;
  s.push(u);
  if (u == root && !head[u])
  {
    dcc[++dccCnt].push_back(u);
    return;
  }
  for (int i = head[u]; i; i = e[i].next)
  {
    int v = e[i].v;
    if (!dfn[v])
    {
      tarjan(v, i);
      low[u] = min(low[u], low[v]);
      if (low[v] >= dfn[u])
      {
        int cur;
        ++dccCnt;
        do
        {
          cur = s.top();
          s.pop();
          dcc[dccCnt].push_back(cur);
        } while (cur != v);
        dcc[dccCnt].push_back(u);
      }
    }
    else
      low[u] = min(low[u], dfn[v]);
  }
}
```

#1272、点双连通分量

题目描述

给定一个 n 个点 m 条边的无向图,求该图中的所有点双连通分量($v-dcc$)

> *你需要使用链式前向星存图,并且尽可能选择节点编号小的点作为 DFS 生成树的根*

输入格式

第一行输入两个整数 n, m

接下来 m 行,每行两个整数 u, v ,表示一条无向边 (u, v)

输出格式

共 $sccCnt$ 行, $sccCnt$ 为点双连通分量数量

对于第 i 行,输出第 i 个点双连通分量的每个点

数据规模

对于 100% 的数据, $n \leq 2 \times 10^4, 1 \leq m \leq 10^5$

输入样例

```
5 6
1 2
2 3
1 3
3 5
4 5
3 4
```

输出样例

```
5 4 3
2 3 1
```

#694、矿场搭建

题目描述

煤矿工地可以看成是由隧道连接挖煤点组成的无向图

为安全起见,希望在工地发生事故时所有挖煤点的工人都能有一条出路逃到救援出口处

于是矿主决定在某些挖煤点设立救援出口,使得无论哪一个挖煤点坍塌之后,其他挖煤点的工人都有一条道路通向救援出口

请写一个程序,用来计算至少需要设置几个救援出口,以及不同最少救援出口的设置方案总数

输入格式

输入包含若干组数据,每组数据的第一行是一个正整数 N ,表示工地的隧道数

接下来的 N 行每行是用空格隔开的两个整数 S 和 T ,表示挖煤点 S 与挖煤点 T 由隧道直接连接

输入数据以 0 结尾。

输出格式

每组数据输出一行

其中第 i 行以 `Case i:` 开始(注意大小写, `Case` 与 `i` 之间有空格, `i` 与 `:` 之间无空格, `:` 之后有空格)
其后是用空格隔开的两个正整数
第一个正整数表示对于第 i 组输入数据至少需要设置几个救援出口
第二个正整数表示对于第 i 组输入数据不同最少救援出口的设置方案总数

输出格式参照以下输入输出样例

数据范围

对于全部的数据 $N \leq 500$,输入数据保证答案小于 2^{64}

#694、矿场搭建

若出口仅由 1 个,若出口塌方将导致无解, 所以出口数量至少为 2

原图可能不连通,各连通块独立考虑根据乘法原理统计方案数

将原图求出 $v-DCC$, 并将 $v-DCC$ 缩点后考虑

由于割点可能属于周围多个 $v-DCC$ 需要将割点独立建点 , 周围 $v-DCC$ 向其连边

逐个考虑各个连通块

- 若无割点
  - 孤立点

    设置一个出口
  - 不为孤立点

    块内设置 2 个出口 , 记连通块大小为 cnt , 方案数贡献为 $\binom{cnt}{2}$

#694、矿场搭建

- 存在割点

  逐个 v – DCC 考虑

  - 缩点后 v – DCC 度为 1

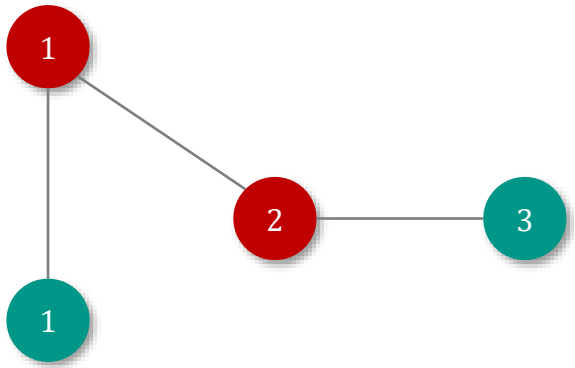
    若割点坍塌 v – DCC 需要一个出口

    此时进需要设置 1 个出口，记 v – DCC 大小为 cnt，方案数贡献为 cnt – 1

  - 缩点后 v – DCC 度 ≥ 2

    若其内点坍塌，所有点可通过另一条路径到达其它 v – DCC

    无需建立出口



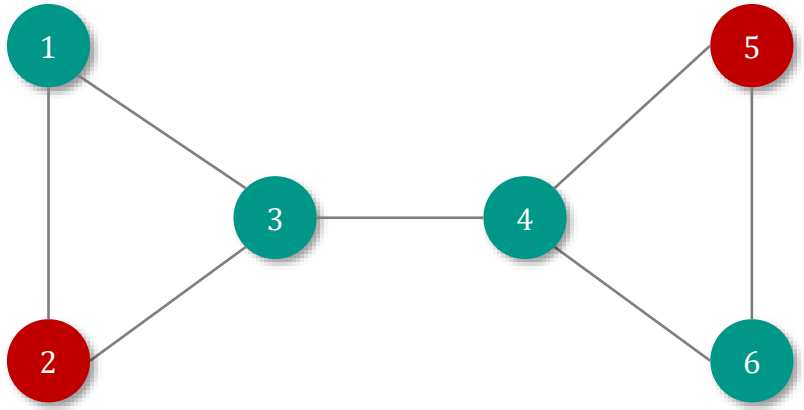
边双连通分量 e-DCC

求出无向图中所有的桥

删除桥后,每个连通块就是一个边双连通分量

用 Tarjan 标记所有的桥边,对整个无向图执行一次 DFS (不访问桥边)

即可划分出每个 e – DCC




```
void tarjan(int u, int eIdx)
{
  dfn[u] = low[u] = ++idx;
  for (int i = head[u]; i; i = e[i].next)
  {
    int v = e[i].v;
    if (!dfn[v])
    {
      tarjan(v, i);
      if (low[v] > dfn[u])
        isBridge[i] = isBridge[i ^ 1] = true;
      low[u] = min(low[u], low[v]);
    }
    else if (i != (eIdx ^ 1))
      low[u] = min(low[u], dfn[v]);
  }
}
void dfs(int u)
{
  dcc[u] = dccCnt;
  for (int i = head[u]; i; i = e[i].next)
  {
    int v = e[i].v;
    if (!dcc[v] && !isBridge[i])
      dfs(v);
  }
}
```

#1291、边双连通分量

题目描述

给定一个 n 个点 m 条边的无向图

求边双连通分量($e-dcc$)数量

输入格式

第一行， n, m

接下来m行，每行两个整数 u, v ，表示一条无向边 (u, v)

输出格式

输出值为边双连通分量数量

数据规模

对于 100% 的数据, $1 \leq n \leq 2 \times 10^4, 1 \leq m \leq 10^5$

输入样例

```
6 7
1 2
2 3
3 1
3 4
4 5
5 6
4 6
```

输出样例

```
2
```

#693、分离的路径

题目描述

为了从 F 个草场中的一个走到另一个，贝茜和她的同伴们不得不路过一些她们讨厌的可怕的树。奶牛们已经厌倦了被迫走某一条路，所以她们想建一些新路，使每一对草场之间都会至少有两条相互分离的路径，这样她们就有多一些选择。

每对草场之间已经有至少一条路径，给出所有 R 条双向路的描述，每条路连接了两个不同的草场，请计算最少的新建道路的数量。

路径由若干道路首尾相连而成，两条路径相互分离，是指两条路径没有一条重合的道路，但是两条分离的路径上可以有一些相同的草场。

对于同一对草场之间，可能已经有两条不同的道路，你也可以在它们之间再建一条道路，作为另一条不同的道路。

输入格式

第一行输入两个整数 F 和 R；

接下来 R 行，每行输入两个整数，表示两个草场，它们之间有一条道路。

输出格式

输出最少需要新建的道路数目。

样例解释

图中虚线表示已有的道路，点线表示新建的两条道路。现在可以检验一些路径，比如：

草场 1 和草场 2：$1 \to 2$ 和 $1 \to 6 \to 5 \to 2$

草场 1 和草场 4：$1 \to 2 \to 3 \to 4$ 和 $1 \to 6 \to 5 \to 4$

草场 3 和草场 7：$3 \to 4 \to 7$ 和 $3 \to 2 \to 5 \to 7$

事实上，每一对草场之间都连接了两条分离的路径。

数据范围与提示

$1 \leq F \leq 5000, F-1 \leq R \leq 10000$。

样例输入

```
7 7
1 2
2 3
3 4
2 5
4 5
5 6
5 7
```

样例输出

```
2
```

#693、分离的路径

给连通的无向图加边,使得无向图没有桥(即变成 e − DCC), 最小化加边数

容易想到 e − DCC 内不存在桥

考虑对整个图求一遍 e − DCC 将 e − DCC 缩为一个点

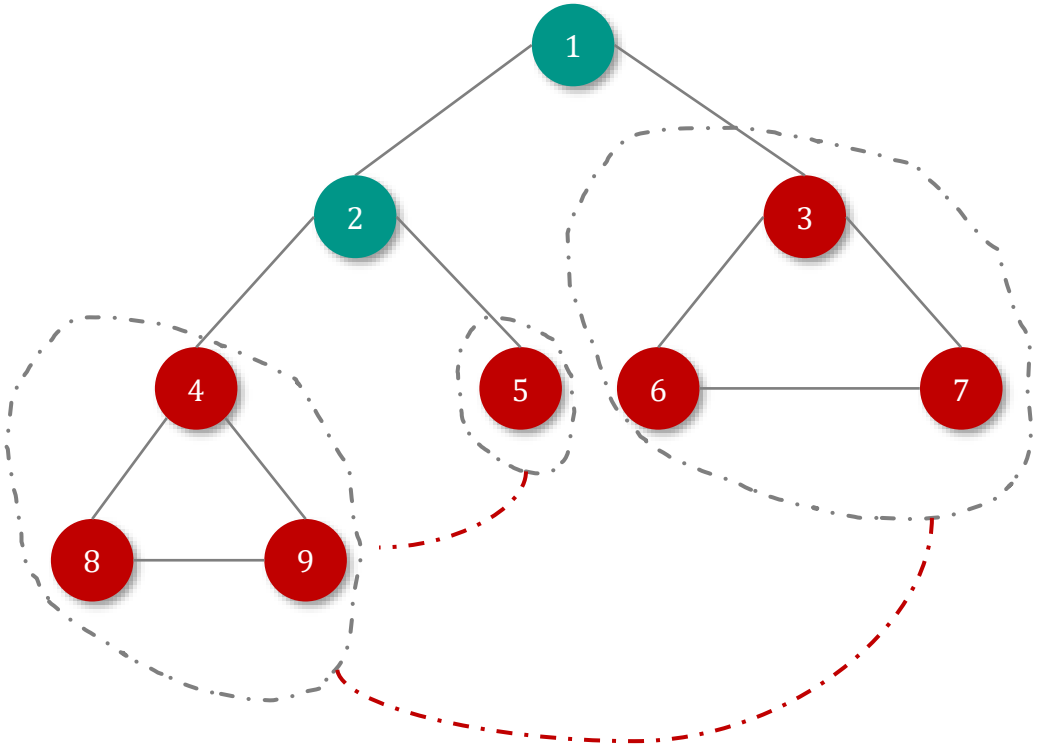
那么缩完点后得到的图一定是一棵树

记缩点后度为 1 的节点数量为 cnt , 所加的边数至少为 $\left\lceil \frac{\text{cnt}}{2} \right\rceil$

对于所有缩点后度为 1 的节点,只需要两两直接连边

那么两点之间简单路径成环 $(u \leftrightarrow \text{LCA}_{u,v} \leftrightarrow v)$

对于 cnt 为奇数时 , 剩下的点向任一点连边即可



谢谢观看