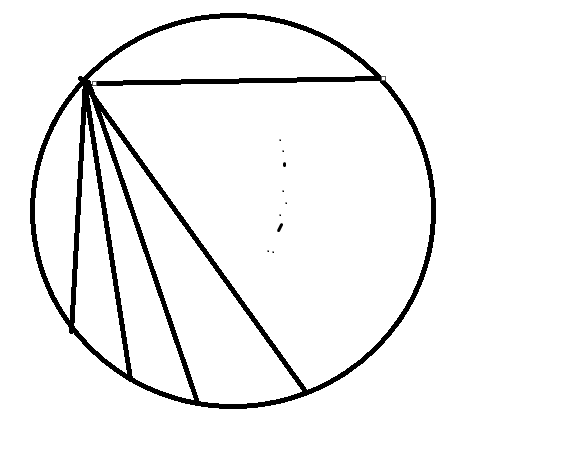
首先无向图中环的数量增长是很快的,像下面这种图只需要少数边和节点就可以让环的数量指数级增长



甚至于,你要计算环的数量,都需要用状压来求,换句话说,找出一个无向图中所有的环是不现实的,如果有这种想法,那一定是错的,除了两种可能:

1.题目保证存在的环很少,比如基环树,找环代码如下(如果找不到就是没有环)

**第一种方法:拓扑排序**处理无向图可以找出环上的所有点。

void topsort()

{

int l = 0, r = 0;

for (int i = 1; i <= n; i++)

if (in[i] == 1)

q[++r] = i;

while (l < r)

{

int now = q[++l];

for (int i = ls[now]; i; i = a[i].next)

{

int y = a[i].to;

if (in[y] > 1)

{

in[y]--;

if (in[y] == 1)

q[++r] = y;

}

}

}

}



**第二种方法:dfs**



inline bool dfs(int x, int in\_edge)

{

if (v[x] == 1)

{ // 再次访问说明到了环的链接点，标记它是环(v[x] = 2)

v[x] = 2, loop[++cnt] = x, v2[x] = 1;

return true;

}

v[x] = 1;

for (int i = head[x]; ~i; i = nex[i])

{

int y = ver[i];

// 如果当前边不是上一条边(没有往回走)并且当前节点在环上

if (i != ((in\_edge) ^ 1) && dfs(y, i))

{ // 每次存一个点，靠继续dfs遍历整个环

if (v[x] != 2) // 当前节点不是衔接点,把环里的点存起来

loop[++cnt] = x, v2[x] = 1, s[cnt] = s[cnt - 1] + edge[i];

else

{ // 环的链接点

s[st - 1] = s[st] - edge[i]; // 又转回来（破环成链）

return false;

}

return true;

}

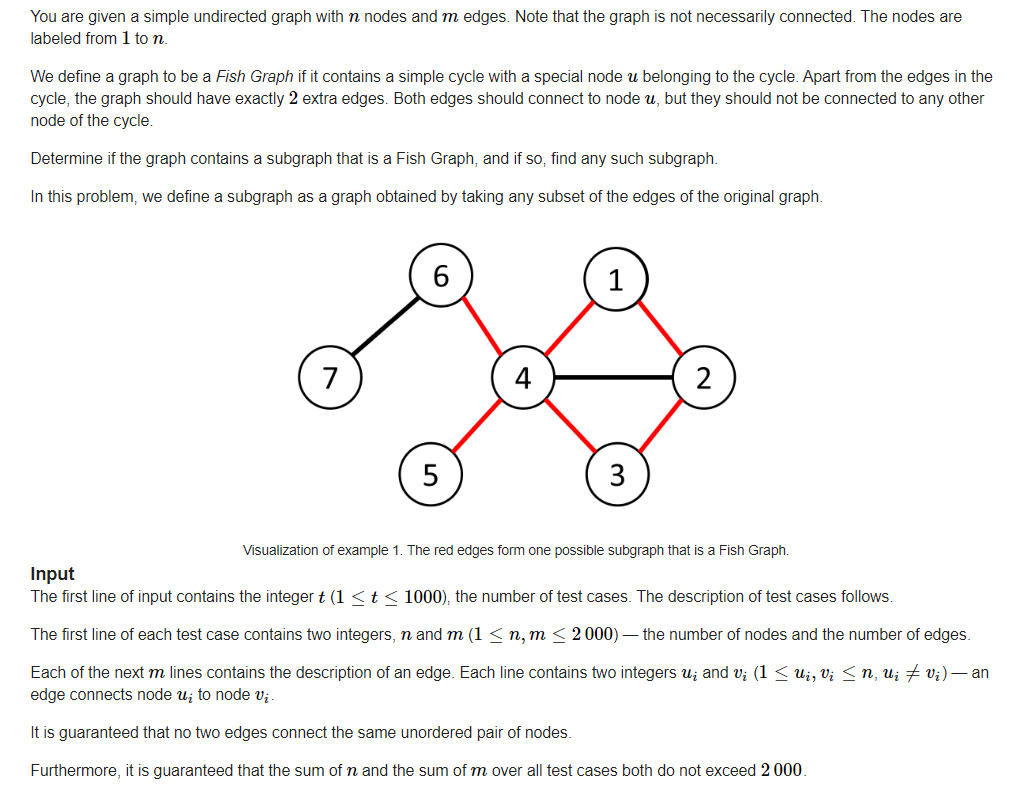
}

return false;

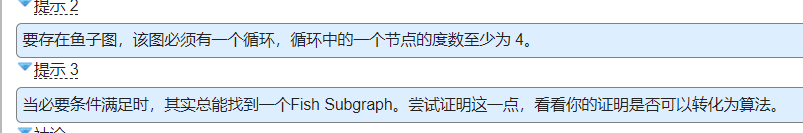
}

2.有可能的是找到经过每个节点的最短的简单环(对每个节点进行bfs遍历即可),比如下题:

<https://codeforces.com/contest/1817/problem/B>



通过推结论可以得到只要有:



那么遍历每个节点再找经过其的简单环,用bfs,就可以得到其是否存在可行方法