ACM 模板

钱智煊，黄佳瑞，车昕宇

2024 年 9 月 13 日

目录

1 图论

1.1 连通性相关

1.1.1 tarjan

```
void tarjan(int x)
{
   dfn[x]=low[x]=++Time,sta[++tp]=x,ins[x]=true;
   for(int i=hea[x];i;i=nex[i])
   {
      if(!dfn[ver[i]])
        ↪ tarjan(ver[i]),low[x]=min(low[x],low[ver[i]]);
      else if(ins[ver[i]])
        ↪ low[x]=min(low[x],dfn[ver[i]]);
   }
   if(dfn[x]==low[x])
   {
      do
      {
         x=sta[tp],tp--,ins[x]=false;
      } while (dfn[x]!=low[x]);
   }
}
```

1.1.2 tarjan 求 LCA

实现均摊 $O(1)$。就是用 tarjan 按照顺序遍历子树的特点加上并查集即可。

```
inline void add_edge(int x,int y){
  ↪ ver[++tot]=y,nex[tot]=hea[x],hea[x]=tot; }
inline void add_query(int x,int y,int d)
    { qver[++qtot]=y,qnex[qtot]=qhea[x],qhea[x]=qtot,
      ↪ qid[qtot]=d; }
int Find(int x){ return (fa[x]==x)?x:(fa[x]=Find(fa[x]));
  ↪ }
void tarjan(int x,int F)
{
    vis[x]=true;
    for(int i=hea[x];i;i=nex[i])
    {
        if(ver[i]==F) continue;
        tarjan(ver[i],x),fa[ver[i]]=x;
    }
    for(int i=qhea[x];i;i=qnex[i])
    {
        if(!vis[qver[i]]) continue;
        ans[qid[i]]=Find(qver[i]);
    }
}
for(int i=1;i<=n;i++) fa[i]=i;
for(int i=1,x,y;i<n;i++)
  ↪ x=rd(),y=rd(),add_edge(x,y),add_edge(y,x);
for(int i=1,x,y;i<=m;i++)
    x=rd(),y=rd(),add_query(x,y,i),add_query(y,x,i);
tarjan(s,s);
for(int i=1;i<=m;i++) printf("%d\n",ans[i]);
```

1.1.3 割点、割边

注意这里的 dfn 表示**不经过父亲**，能到达的最小的 dfn 。割点：

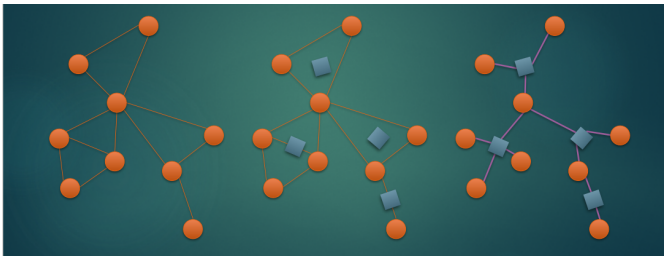
- 若 u 是根节点，当至少存在 2 条边满足 $low(v) \geq dfn(u)$ 则 u 是割点。
- 若 u 不是根节点，当至少存在 1 条边满足 $low(v) \geq dfn(u)$ 则 u 是割点。

割边：

- 当存在一条边条边满足 $low(v) > dfn(u)$ 则边 i 是割边。

注意：记录上一个访问的边时要记录边的编号，不能记录上一个过来的节点（因为会有重边）！！！或者在加边的时候特判一下，不过注意编号问题。（用输入顺序来对应数组中位置的时候，重边跳过，但是需要 tot+=2）

1.1.4 圆方树



圆方树会建立很多新的点，所以不要忘记**给数组开两倍**！

```
void tarjan(int u)
{
   dfn[u]=low[u]=++Time,sta[++tp]=u;
   for(int v:G[u])
   {
      if(!dfn[v])
      {
         tarjan(v),low[u]=min(low[u],low[v]);
         if(low[v]==dfn[u])
         {
            int hav=0; ++All;
            for(int x=0;x!=v;tp--)
              ↪ x=sta[tp],T[x].pb(All),T[All].pb(x),hav++;
            T[u].pb(All),T[All].pb(u),;
            siz[All]=++hav;
         }
      }
      else low[u]=min(low[u],dfn[v]);
   }
}
```

1.2 同余最短路

形如：

- 设问 1：给定 n 个整数，求这 n 个整数在 $h(h \leq 2^{63}-1)$ 范围内 **能拼凑出多少的其他整数（整数可以重复取）**。
- 设问 2：给定 n 个整数，求这 n 个整数 **不能拼凑出的最小（最大）的整数**。

设 x 为 n 个数中最小的一个，令 $ds[i]$ 为只通过增加其他 $n-1$ 种数能够达到的最低楼层 p ，并且满足 $p \equiv i \pmod{x}$ 。

对于 $n-1$ 个数与 x 个 $ds[i]$ ，可以如下连边：

```
for(int i=0;i<x;i++) for(int j=2;j<=n;j++)
  ↪ add(i,(i+a[j])%x,a[j]);
```

之后进行最短路，对于：

- 问在 h 范围内能够到达的**点的数量**：答案为（加一因为 i 本身也要计算）：

$$\sum_{i=0}^{x-1} [d[i] \leq h] \times \frac{h-d[i]}{x} + 1$$

- 问不能达到的**最小的数**：答案为（i 一定时最小表示的数为 $d[i]=s\times x+i$，则 $(s-1)\times x+i$ **一定不能**被表示出来）：

  注意：ds 与 h 范围相同，一般也要开 long long！

$$\min_{i=1}^{x-1}\{d[i]-x\}$$