连通分量

强连通分量

只有有向图才有这种定义，强连通分量中，点两两可达。

SCC 主要用于缩点，每一个点仅属于一个 SCC，缩点后是一个 DAG，在建新图过程中需要注意两点不应属于同一 SCC。

在实现中，在遍历到 $p\rightarrow q$ 返祖边时，对于 `low[p]` 的更新，`min(low[p],low[q])` 和 `min(low[p],dfn[q])` 等价，均可正确计算出结果。这与点双连通分量是不同的。

模板题：缩点

```
int sta[N],top;
int col[N],num;
int dfn[N],low[N],tot;
int vis[N];
void tarjan(int p){
    dfn[p]=low[p]=++tot;
    int i,q;
    sta[++top]=p;vis[p]=1;
    for(i=hd[p];i;i=e[i].n){
        q=e[i].e;
        if(!dfn[q])tarjan(q),low[p]=min(low[p],low[q]);
        else if(vis[q])low[p]=min(low[p],low[q]);
    }
    if(dfn[p]==low[p]){
        num++;
        while(sta[top+1]!=p){
            q=sta[top];
            col[q]=num;
            vis[q]=0;
            top--;
        }
    }
}
```

点双连通分量

一个点双删掉任意点后图仍然连通。注意，根据此定义，K_1 和 K_2 均为点双。

v-DCC 主要用于判割点，每一个非割点仅属于一个 v-DCC，割点可以同时属于多个 v-DCC。

在实现中，如果某个点 p 是割点，它应当满足以下条件之一：

1. p 是根节点，且以它为根的 dfs 树有大于一个分支；
2. p 不是根节点，且它的某个孩子 q 不能返祖（即 low[q]>=dfn[p] □□

由于求 DCC 是通过某个有孩子的割点进行判定的，故需要对孤立点进行特判。

模板题：点双 割点

```
int low[N],dfn[N],tot;
int sta[N],top;
int cut[N];
vector<int>dcc[N];int dcc_cnt;
void tarjan(int p,int f){
    int q,i,child=0;
    sta[++top]=p;
    dfn[p]=low[p]=++tot;
    for(i=hd[p];i;i=e[i].n){
        q=e[i].e;
        if(q==f)continue; // avoid return to fa
        if(!dfn[q]){
            tarjan(q,p);
            low[p]=min(low[p],low[q]);
            if(low[q]>dfn[p]);// i is cut-edge
            if(low[q]>=dfn[p]){
                // p is cut of branch q
                if(f!=-1)cut[p]=1;
                dcc[++dcc_cnt].pb(p);
                int r;
                do{
                    r=sta[top--];
                    dcc[dcc_cnt].pb®;
                }while(r!=q);
            }
            child++;
        }
        low[p]=min(low[p],dfn[q]);
    }
    if(f==-1){ // edge case: p is root
        if(child>1)cut[p]=1; // 2 branch in dfs tree
        if(hd[p]==0)dcc[++dcc_cnt].pb(p); // single point
    }
}
```

边双连通分量

一个边双删掉任意边后图仍然连通。

无向图中的 e-DCC 和有向图中的 SCC 很相似，有的时候可以进行缩点。

在实现中 e-DCC 和 SCC 的实现方式也类似，只是加了一个不能走父边的限制。

模板题：边双

```
struct Edge{
    int e,n;
}e[N«1];
int hd[N],cnt=1,n,m;
void add(int a,int b){
    e[++cnt].e=b;e[cnt].n=hd[a];hd[a]=cnt;
}
int sta[N],top;
int col[N],num;
int dfn[N],low[N],tot;
int vis[N];
int ban[M«1];
void tarjan(int p){
    dfn[p]=low[p]=++tot;
    int i,q;
    sta[++top]=p;vis[p]=1;
    for(i=hd[p];i;i=e[i].n){
        q=e[i].e;
        if(ban[i])continue;
        if(!dfn[q]){
            ban[i]=ban[i^1]=1;
            tarjan(q),low[p]=min(low[p],low[q]);
        }
        else if(vis[q])low[p]=min(low[p],low[q]);
    }
    if(dfn[p]==low[p]){
        num++;
        while(sta[top+1]!=p){
            q=sta[top];
            col[q]=num;
            vis[q]=0;
            top—;
        }
    }
}
```

杂项

割点和割边没有任何联系，除了他们姓一样：

割边两段不一定是割点：

两割点不一定组成割边：