信奥算法

快快编程2474

```
 8      cin>>n;
 9      for(int i=0;i<n;++i)cin>>x[i];
10      int ID;
11      for(int i=0;i<n;++i)if(x[i]==1){
12          ID=i;
13          break;
14      }
15      int LID=(ID==0?n:ID)-1;
16      int RID=(ID==n-1?-1:ID)+1;
17      int delta;
18      if(                  ) delta=-1;
19      else delta=1;
20      cout<<x[ID];
21      for(int i=1;i<n;++i){
22          
23          if(ID<0) ID=n-1;
24          else if(      ) ID=0;
25          cout<<" "<<x[ID];
26      }
```

快快编程2475

无向边

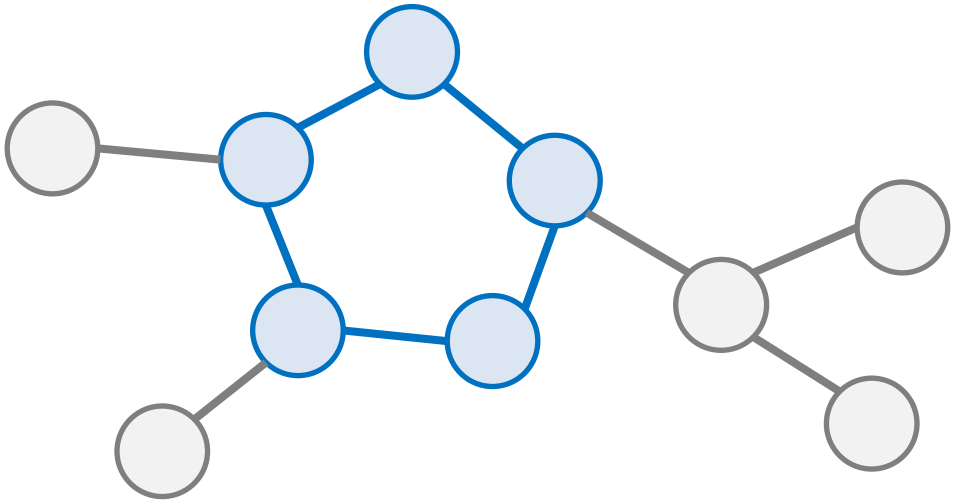
可能不连通
形成基环森林

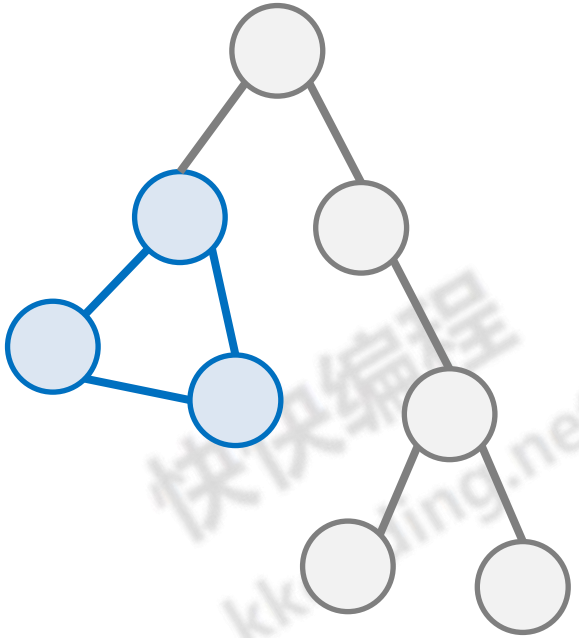
无向基环树

别称：环套树/树套环

n个节点+n条无向边

若连通
恰比树多1边





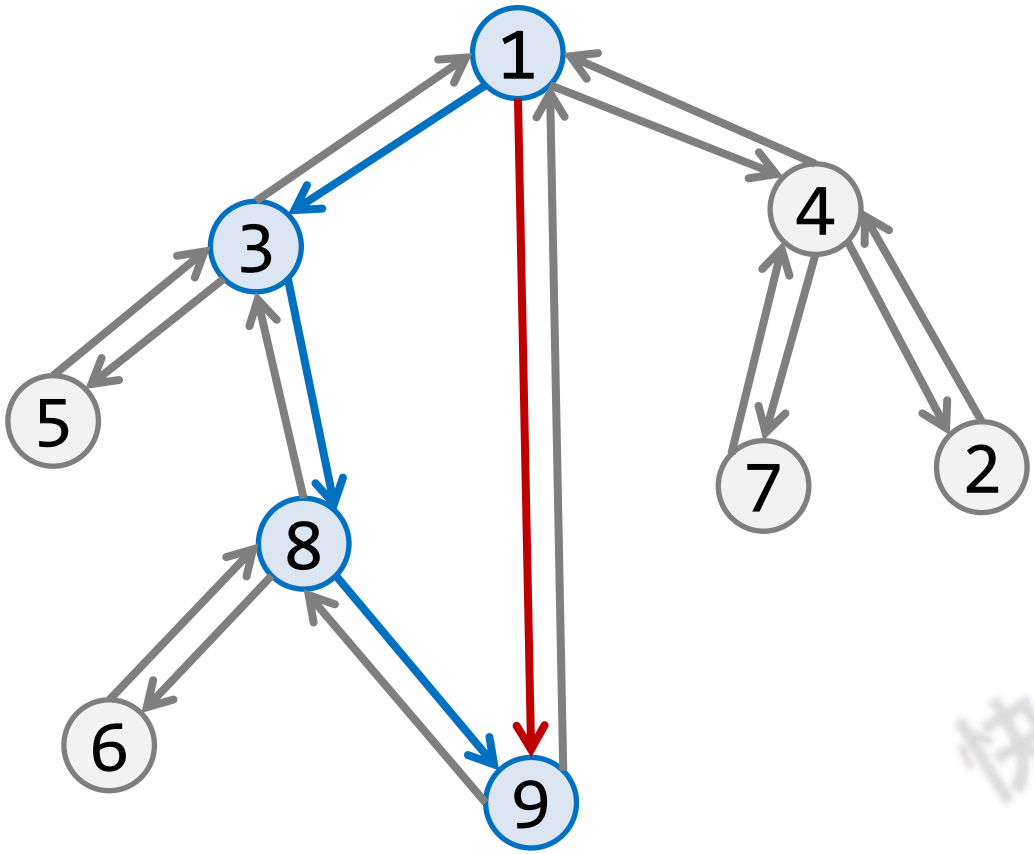
无向边

难点:
无向边
变成
双向边

特点:
祖先指向
已访问过
的子孙

识别唯一一条非树边

识别非树边对应的
双向边中的1条



利用DFS
遍历过程
顺便发现环

```
 9  struct edge{int to,nxt;};
10  edge e[N*2];
11  void add(int u,int v){
12      e[++nE]=(edge){v,hd[u]};
13      hd[u]=nE;
14  }
```

```
51      cin>>n;
52      for(int i=1;i<=n;++i){
53          int x; cin>>x;
54          add(i,x); add(x,i);
55      }
56      for(int i=1;i<=n;++i)
57          if(!dfn[i])
58              dfs(i,0);
59      cout<<nC<<endl;
60      for(int k=1;k<=nC;++k)
61          print(circle[k]);
```

关键边特点:
祖先指向
已访问子孙

```
15 void dfs(int u,int fa){
16     p[u]=fa;
17     dfn[u]=++timer;
18     for(int i=hd[u];i;i=e[i].nxt){
19         int v=e[i].to;
20         if(!dfn[v]){dfs(v,u);continue;}
21         if(v==fa)continue;
22         if(dfn[v]<dfn[u])continue;
23         nC++;
24         circle[nC].push_back(u);
25         for(;v!=u;v=p[v])
26             circle[nC].push_back(v);
27     }
28 }
```

能否删除第21行

关键边特点:
祖先指向
已访问子孙

```
15 void dfs(int u,int fa){
16     p[u]=fa;
17     dfn[u]=++timer;
18     for(int i=hd[u];i;i=e[i].nxt){
19         int v=e[i].to;
20         if(!dfn[v]){dfs(v,u);continue;}
21         if(dfn[v]<dfn[u])continue;
22         nC++;
23         circle[nC].push_back(u);
24
25
26     }
27 }
```

```
28 void print(vector<int> &C){
29     int len=C.size();
30     int ID=0;
31     for(int i=0;i<len;++i)
32         if(C[i]<C[ID])
33     cout<<C[ID];
34     int LID=(ID==0?len:ID)-1;
35     int RID=                    +1;
36     int delta;
37     delta=(C[LID]<C[RID]?-1:1);
38     for(int i=1;i<len;++i){
39
40         if(ID==-1)ID=len-1;
41         else if(      )ID=0;
42         cout<<" "<<C[ID];
43     }
44     cout<<endl;
45 }
```

快快编程2476

有向边

有向基环树

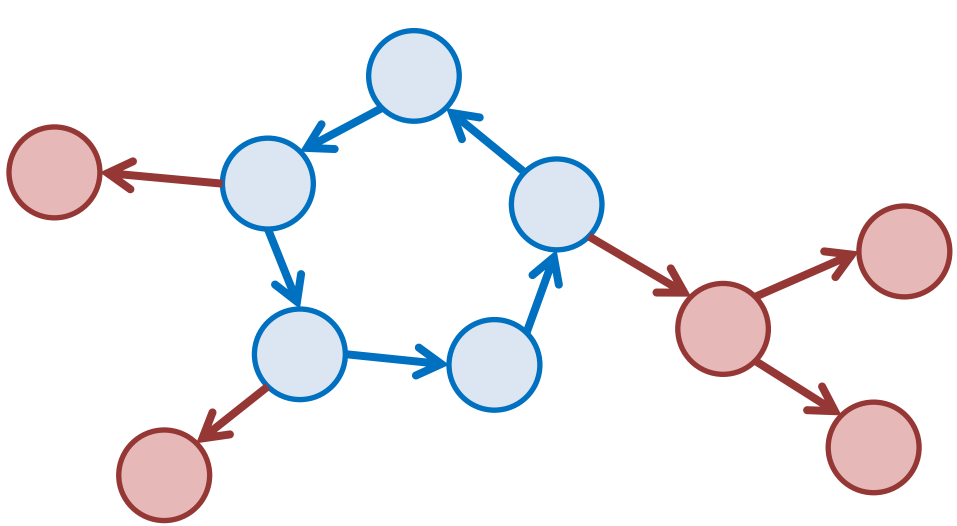
外向树	每个节点都被另1个节点指向
内向树	每个节点都有1个指向目标

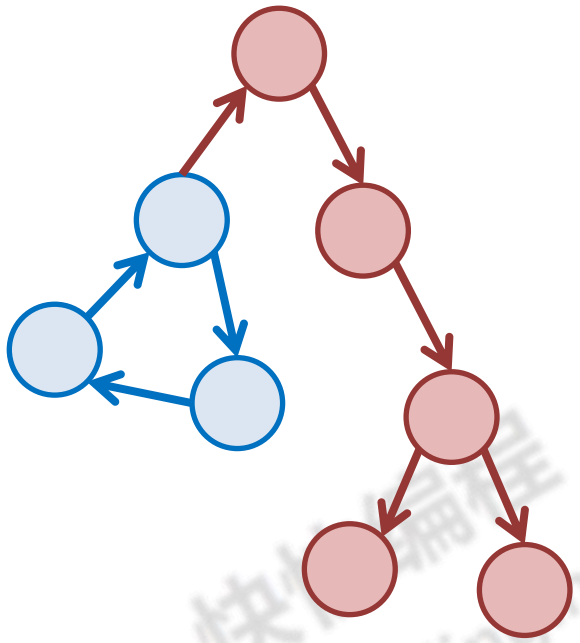
有向边

可能不连通
形成基环森林

有向基环树

外向树	每个节点都被另1个节点指向





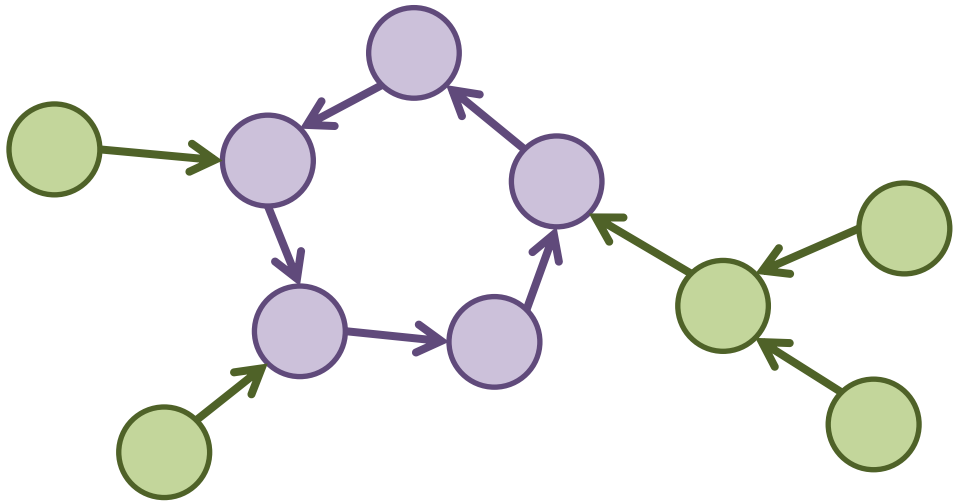
有向边

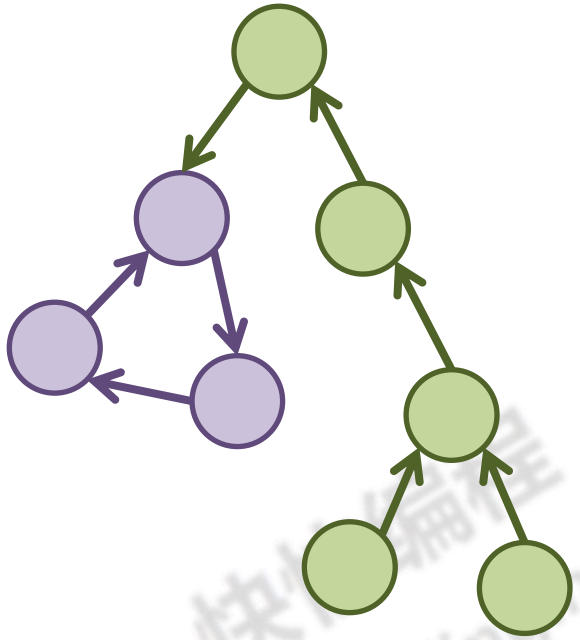
可能不连通
形成基环森林

有向基环树

内向树

每个节点都有1个指向目标





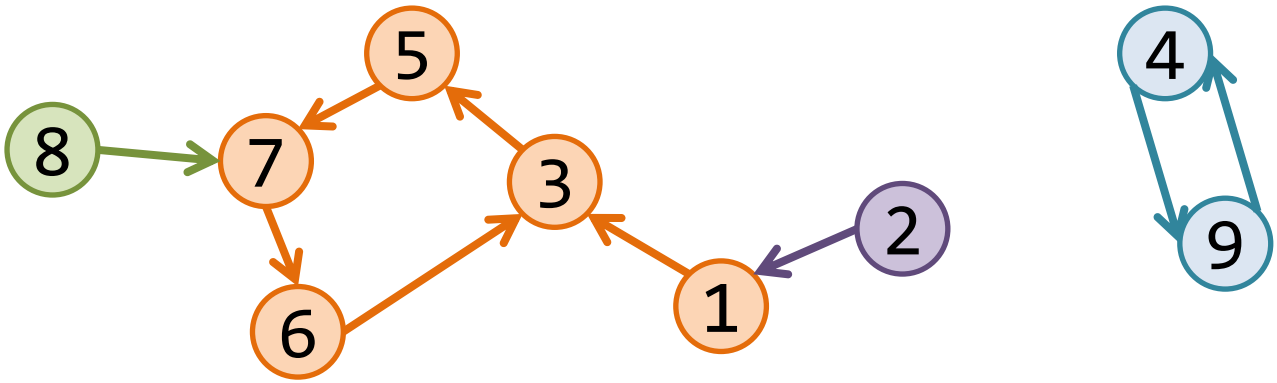
外向情况可以转换为内向情况

沿着内向边逐步发现环！

有向边

内向树

有向基环森林找环



正确理解	DFS访问到本轮DFS刚访问过节点v时，v一定在环上

有向基环森林找环

```
44      cin>>n;
45      for(int i=1;i<=n;++i){
46          cin>>x[i];
47          add(i,x[i]);
48      }
49      for(int i=1;i<=n;++i)
50          if(!dfn[i]){
51              ++nGroup;
52              dfs(i);
53          }
54      cout<<nC<<endl;
55      for(int k=1;k<=nC;++k)
56          print(circle[k]);
```

nGroup: 当前有几组节点

有向基环森林找环

```
16 void dfs(int u){
17     dfn[u]=++timer;
18     iGrp[u]=nGroup;
19     for(int i=hd[u];i;i=e[i].nxt){
20         int v=e[i].to;
21         if(!dfn[v]){dfs(v);continue;}
22         if(iGrp[v]<iGrp[u])continue;
23         nC++;
24         circle[nC].push_back(u);
25         for(;v!=u;v=x[v])
26             circle[nC].push_back(v);
27     }
28 }
```

iGrp[u]: u属于第几组

发现	每个节点只有唯一后继 不需要邻接表/前向星

```
29      cin>>n;
30      for(int i=1;i<=n;++i)
31          cin>>x[i];
32      for(int i=1;i<=n;++i)
33          if(!dfn[i]){
34              ++nGroup;
35              move(i);
36          }
37      cout<<nC<<endl;
38      for(int k=1;k<=nC;++k)
39          print(minCircleID[k]);
```

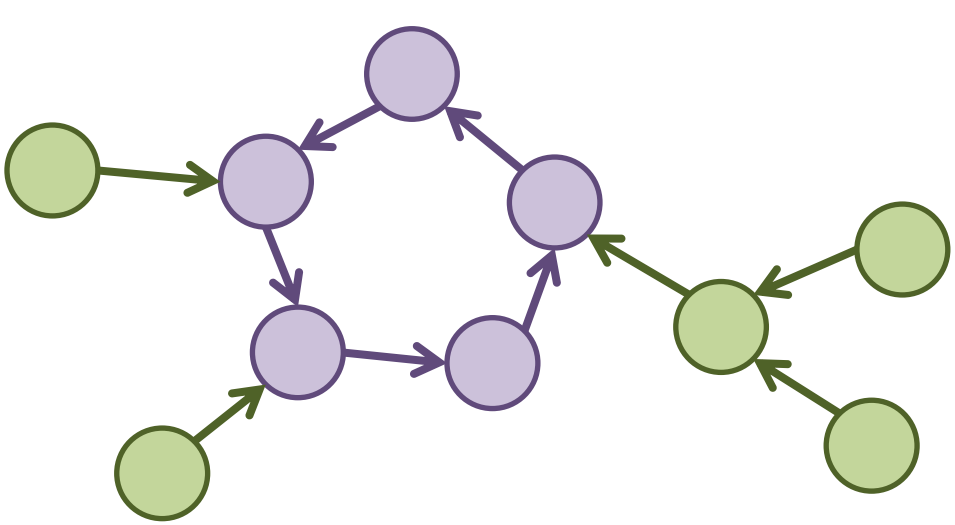
```
 9  void move(int u){
10      dfn[u]=++timer;
11      iGrp[u]=nGroup;
12      int v=x[u];
13      if(!dfn[v]){move(v);return;}
14      if(iGrp[v]<iGrp[u])return;
15      nC++;
16      minCircleID[nC]=u;
17      for(;v!=u;v=x[v])
18          minCircleID[nC]=min(minCircleID[nC],v);
19  }
```

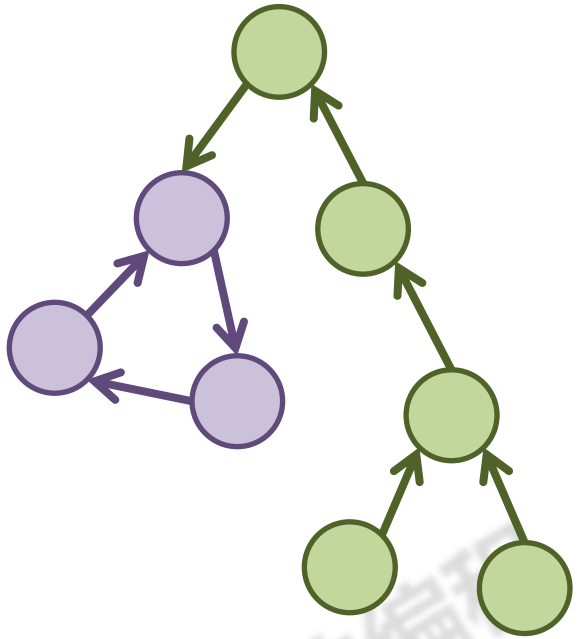
```
20 void print(int u){
21     cout<<u;
22     for(int v=x[u];                )
23         cout<<" "<<v;
24     cout<<endl;
25 }
```

```
37     cout<<nC<<endl;
38     for(int k=1;k<=nC;++k)
39         print(minCircleID[k]);
```

快快编程648

内向 有向基环森林求最小环长度





内向	有向基环森林求最小环长度

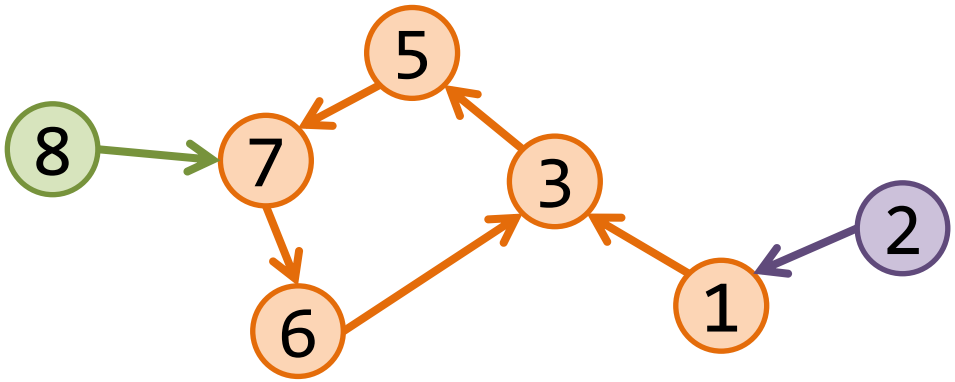
实现1	递归版本实现找环

实现2	循环版本实现找环

```
 7 void move(int u){
 8     dfn[u]=++timer;
 9     iGrp[u]=nGroup;
10     int v=t[u];
11     if(!dfn[v]){                    }
12     if(iGrp[v]==iGrp[u])
13         ans=min(ans,                    );
14 }
```

```
18     cin>>n;
19     for(int u=1;u<=n;++u)cin>>t[u];
20     ans=n;
21     for(int u=1;u<=n;++u)
22         if(!dfn[u]){
23             ++nGroup;
24             move(u);
25         }
26     cout<<ans<<endl;
```

```
 7  void move(int u){
 8      int v=u;
 9      do{
10          dfn[v]=++timer;
11          iGrp[v]=nGroup;
12          [          ]
13      }while([          ]);
14      if(iGrp[v]==iGrp[u])
15          ans=min(ans,[          ]);
16  }
```



快快编程作业

2475

2476

648