C++算法

快快编程1962

解法思路的形成
如何启发思路

A 暴力

B 简化为链

C 手算样例

解法思路的形成
如何启发思路

A 暴力

B 简化为链

C 手算样例

如何暴力枚举

暴力方法1

枚举树上所有 n个点中的每一个节点x
判断x是否在[u,v]路径上 判断x是否在[a,b]路径上

暴力方法2

枚举路径[u,v]上的 每一个节点x
判断x是否在[a,b]路径上

暴力方法1

枚举所有节点

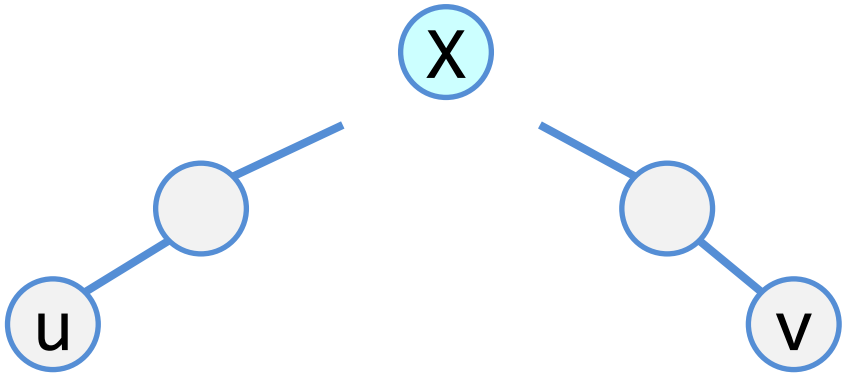
```
53      int ans=0;
54      scanf("%d",&q);
55      for(int i=1;i<=q;++i){
56          int u,v,a,b;
57          scanf("%d %d %d %d",&u,&v,&a,&b);
58          int uv=dst(u,v);
59          int ab=dst(a,b);
60          for(int x=1;x<=n;++x)
61              判断x是否在[u,v]路径上
62              判断x是否在[a,b]路径上 {
63                  ++ans;
64                  break;
65              }
66      }
67      printf("%d\n",ans);
```

暴力方法2

枚举路径[u,v]上的
每一个节点x

判断x是否在[a,b]路径上

路径[u,v]分为3部分
分段枚举



暴力方法2

枚举路径[u,v]上的
每一个节点x

判断x是否在[a,b]路径上

```
scanf("%d %d %d %d",&u,&v,&a,&b);
int uv=lca(u,v);
int dab=dst(a,b);
if(dab==dst(a,uv)+dst(b,uv)){++ans;continue;}
bool OK=0;
while(u!=uv){
    if(dab==dst(a,u)+dst(b,u)){++ans;OK=1;break;}
    u=p[u][0];
}
if(OK)continue;
```

解法思路的形成
如何启发思路

A 暴力

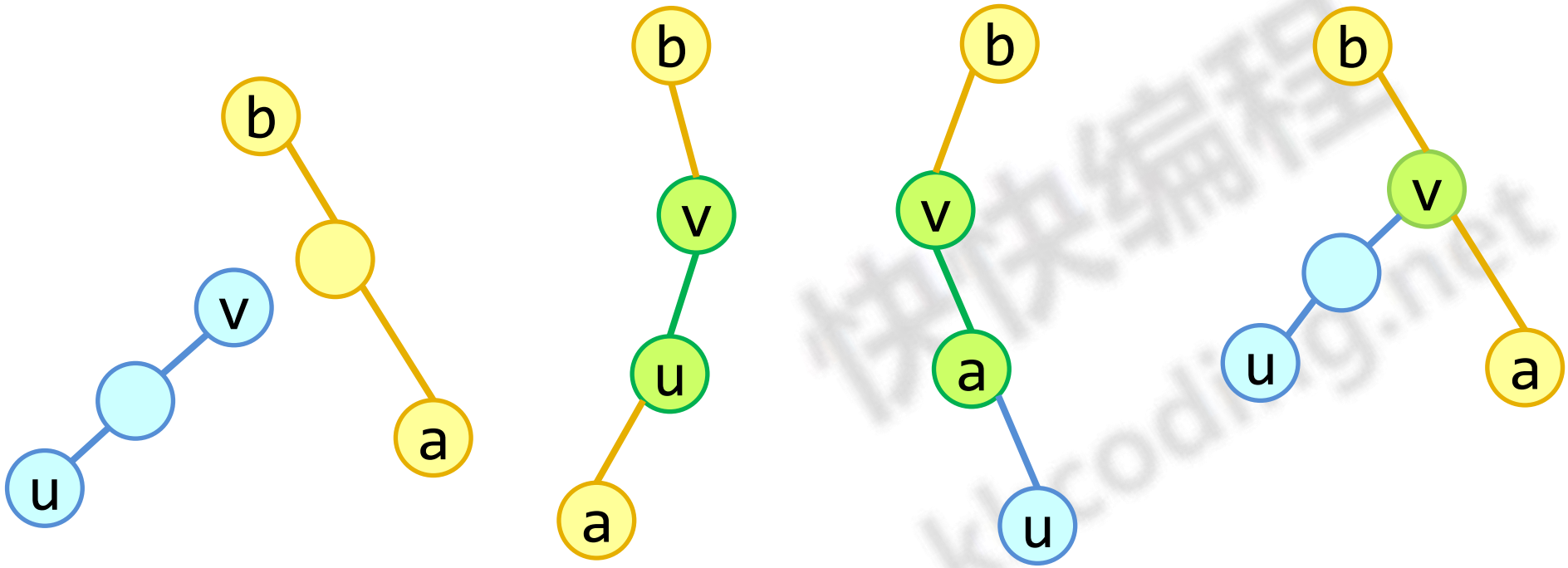
B 简化为链

C 手算样例

简化为链

链状解法

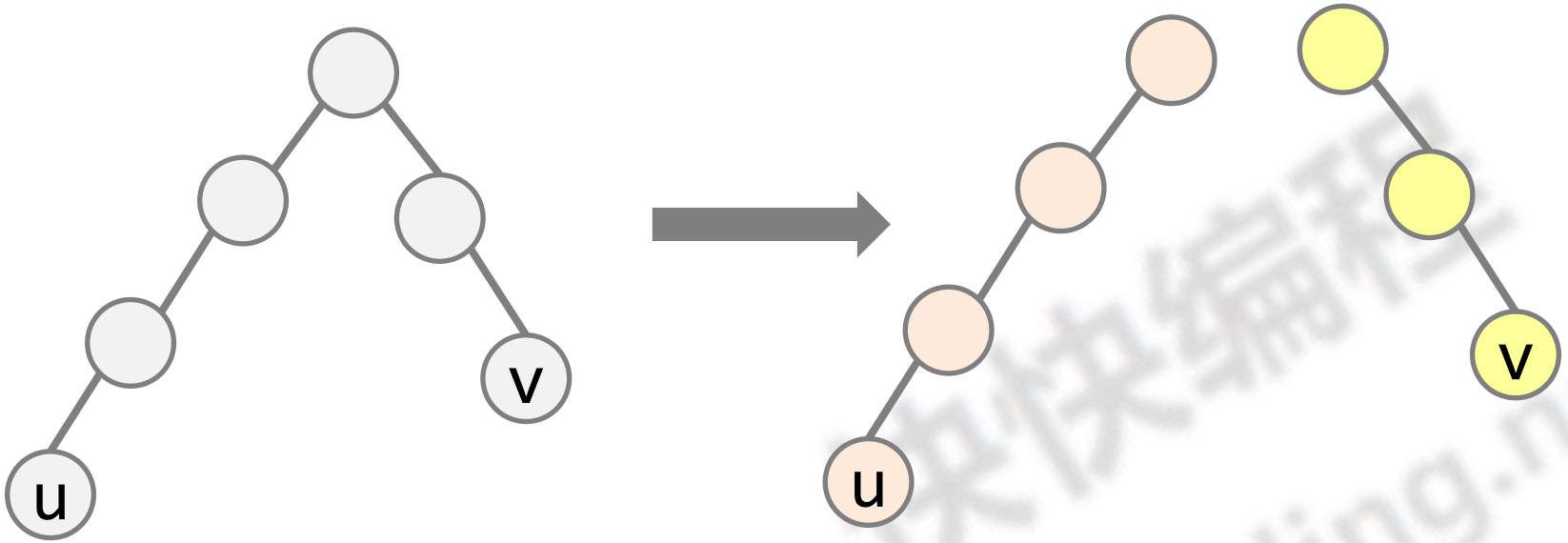
直线链条[u,v]
直线链条[a,b]
几种可能关系：
分离，完全包含，部分重叠



简化为链

路径拆成链

路径[u,v]可以
分为2条直线型链条
分段处理



路径[a,b]也可以
分为2条直线型链条

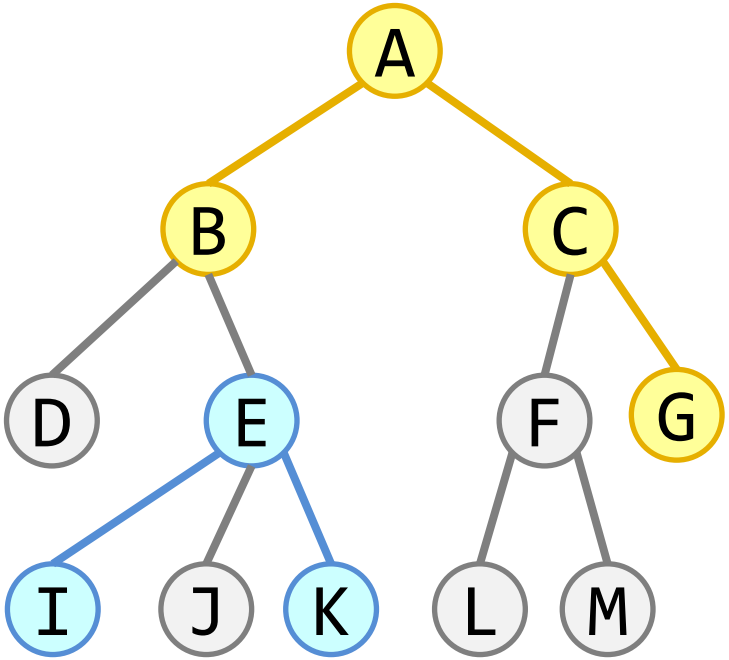
解法思路的形成
如何启发思路

A 暴力

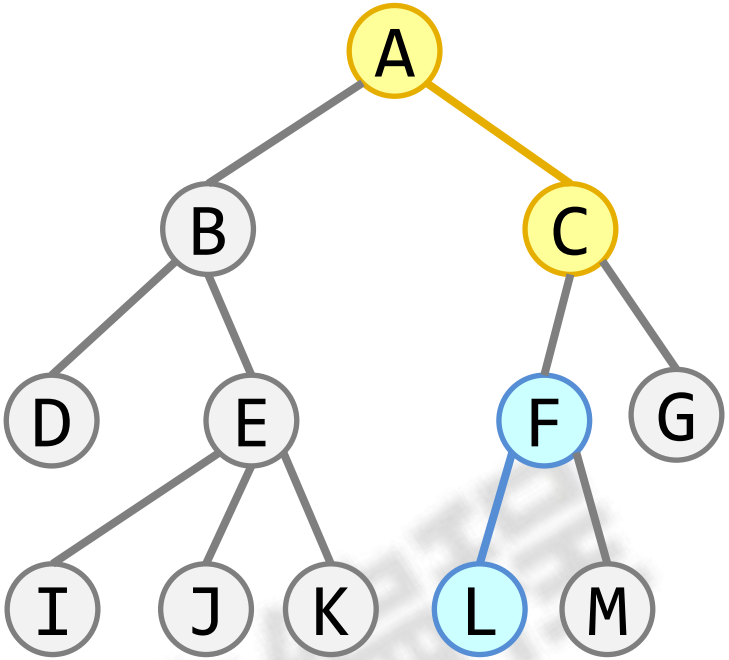
B 简化为链

C 手算样例

手算
启发
灵感

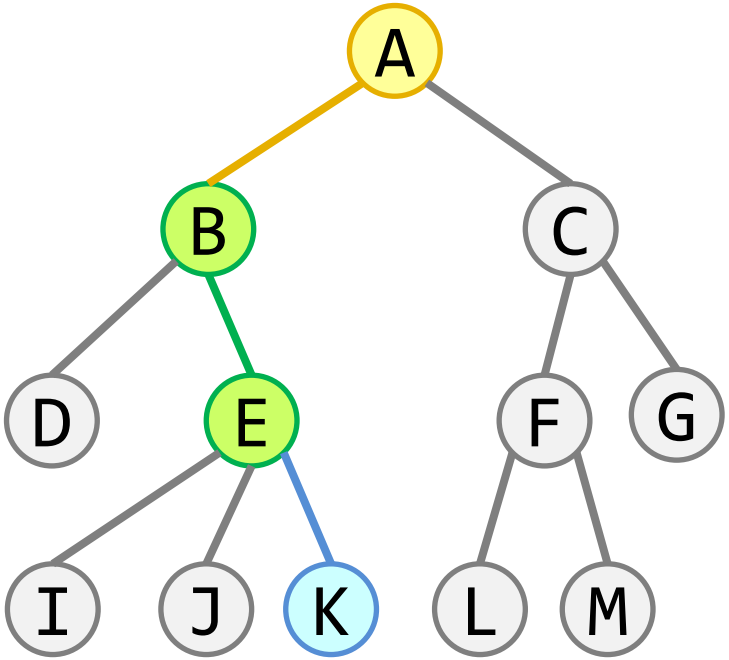


[I,K],[B,G]
不相交

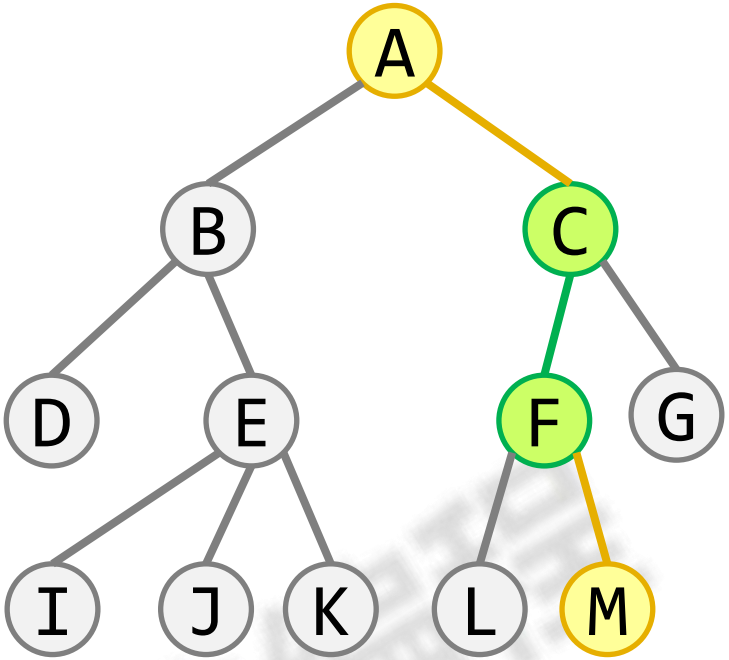


[L,F],[A,C]
不相交

手算
启发
灵感

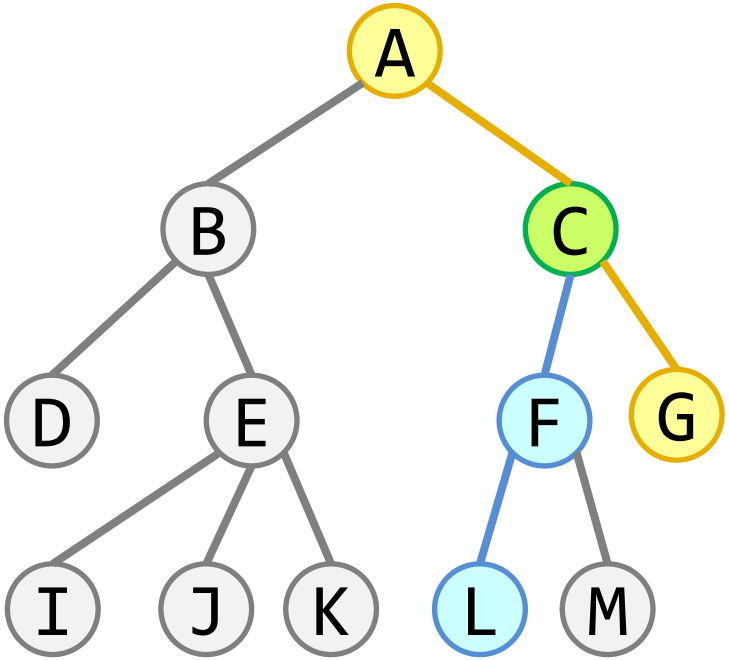


[B,K],[A,E]
相交

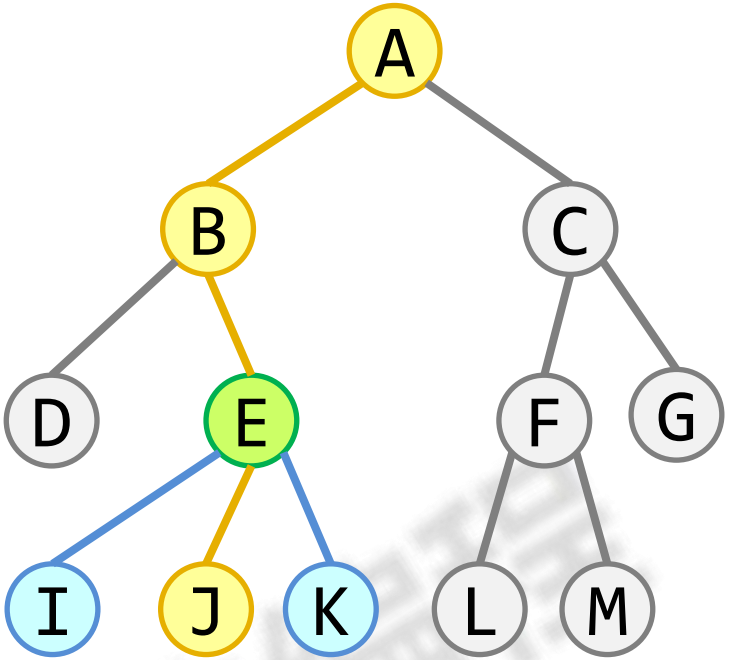


[C,F],[A,M]
相交

手算
启发
灵感



[L,C],[A,G]
相交

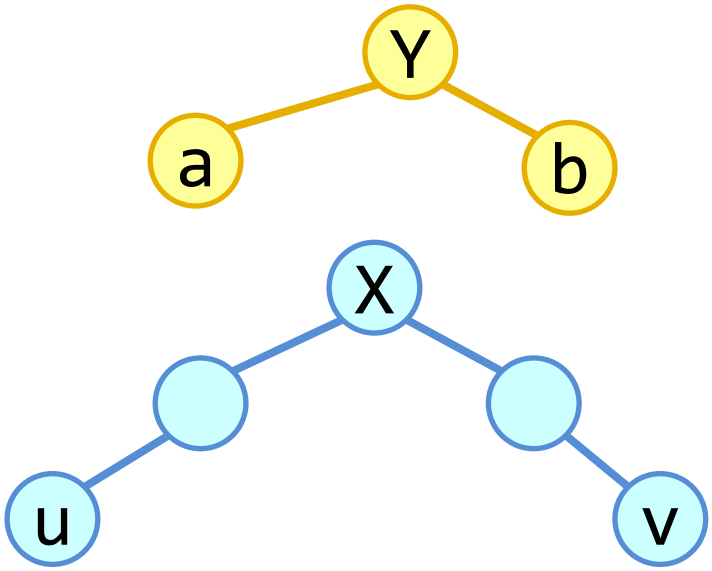


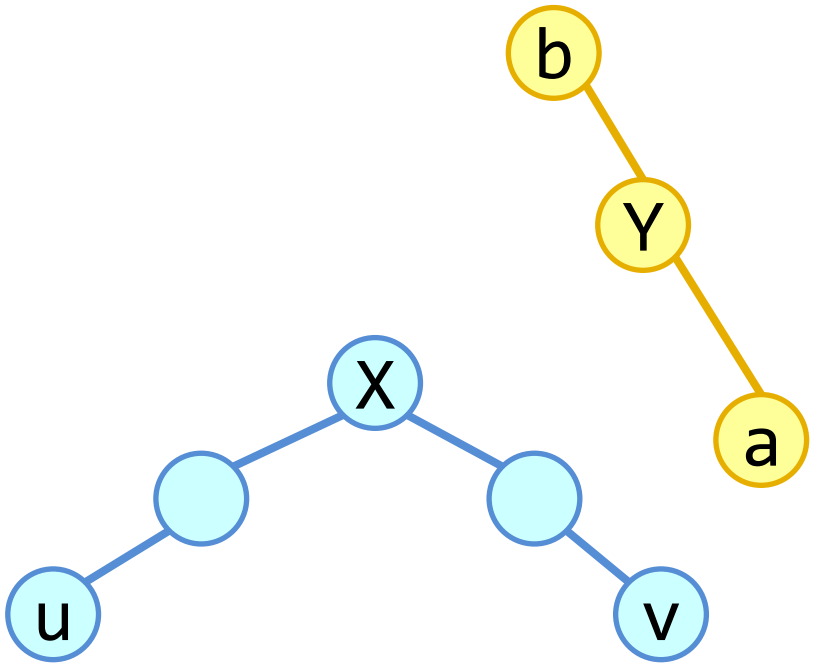
[I,K],[A,J]
相交

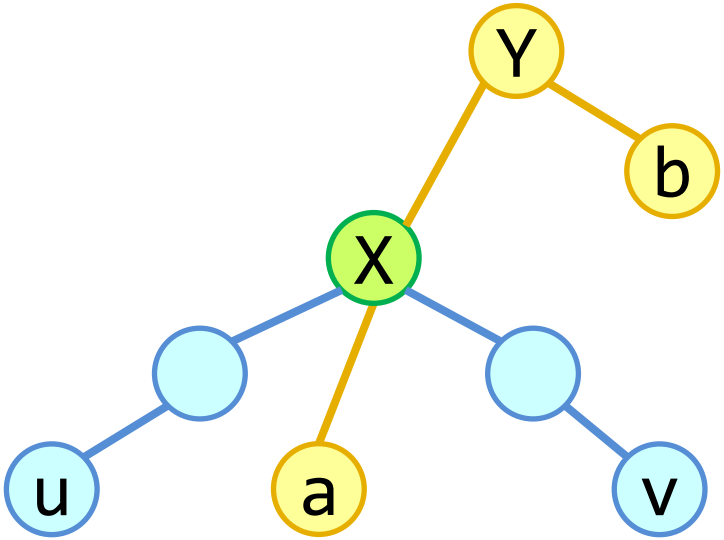
分类讨论

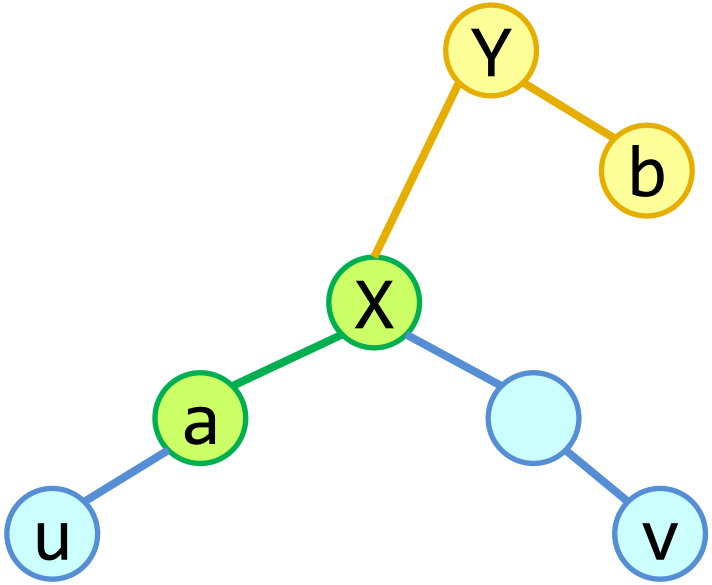
简单类别	2条路径都是"直线型"
复杂类别	有1条路径是"折线型" 有1条路径是"直线型"
	2条路径都是"折线型"
思考	LCA(u,v)和LCA(a,b)的关系

不妨设LCA(u,v)
不高于LCA(a,b)


Y
a
b
X
u
v

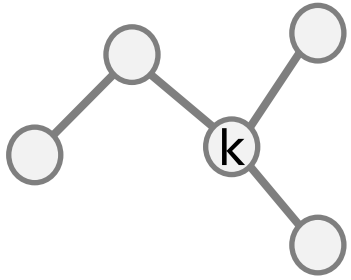

b
Y
a
X
u
v


Y
b
X
u
a
v

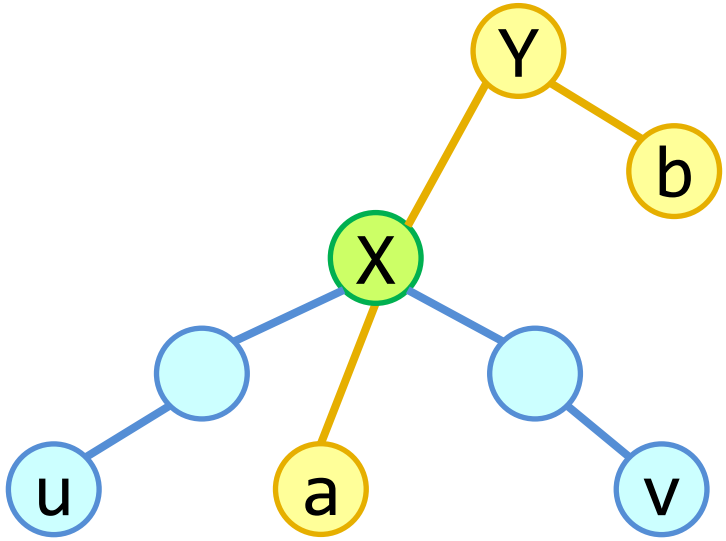

Y
b
X
a
u
v

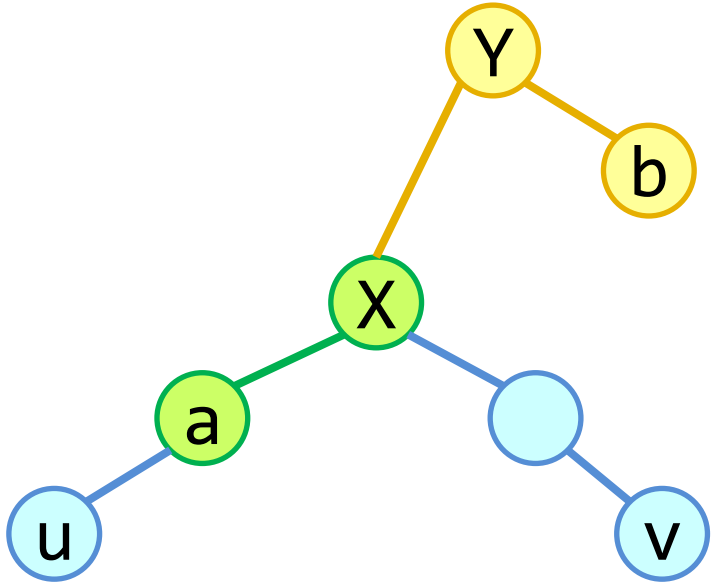
不妨设LCA(u,v)
不高于LCA(a,b)

重大发现	[a,b]要和[u,v]相交 必须经过LCA(u,v)
提出猜想	若两条路径相交 必经过较低的LCA



以上图形不可能!
k最多一个父节点





较低LCA

```
39  bool onPath(int x,int u,int v){
40      return dst(u,x)+dst(v,x)==dst(u,v);
41  }

53      int ans=0;
54      scanf("%d",&q);
55      for(int i=1;i<=q;++i){
56          int u,v,a,b;
57          scanf("%d %d %d %d",&u,&v,&a,&b);
58          int uv=lca(u,v);
59          int ab=lca(a,b);
60          if(                    )
61              ans+=                          
62          else
63              ans+=onPath(uv,a,b);
64      }
```

树的直径

diameter

两点间最长的距离

树的直径

方法1

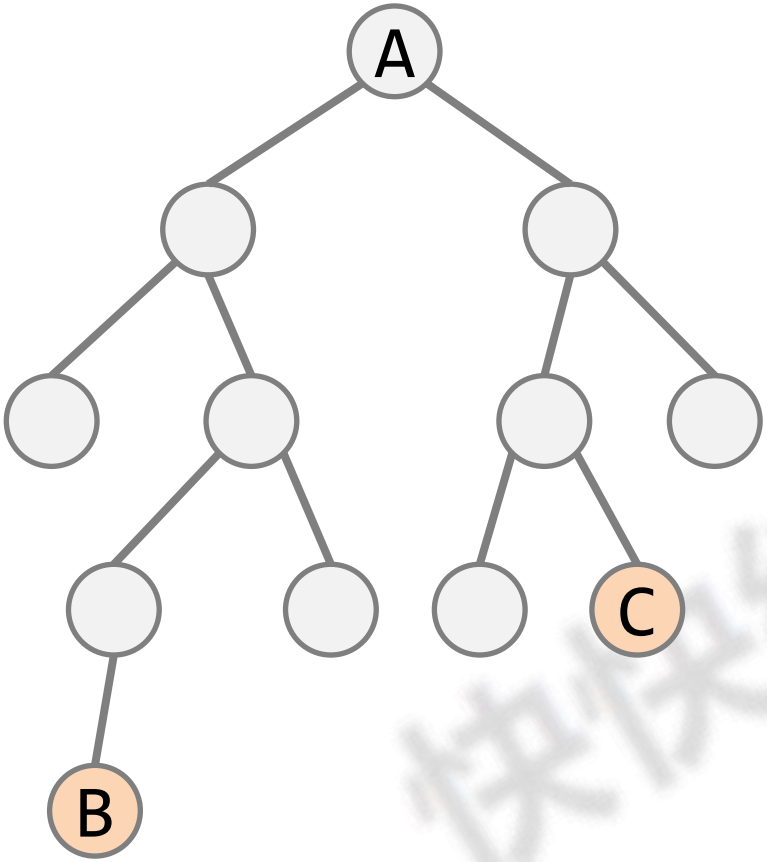
枚举路径转折点u
对每个节点u预计算 从u出发到其子树内 不重叠的最长路和次长路 h[u][1],h[u][2]

方法2

直径的2个端点 一定是"边缘点"	如何证明
2次DFS/BFS 每次找出1个端点	

树直径

直径的2个端点
一定是"边缘点"



树直径	直径的2个端点 一定是"边缘点"
第1次DFS	以任意点A为根节点DFS 找到距离A最远的点B
	一定存在直径包含B
第2次DFS	以B为根节点DFS 找到距离B最远的点C
	BC一定形成直径

树直径

直径的2个端点
一定是"边缘点"

第1次DFS

第2次DFS

```
dfs(1,0);
int id=max_element(d+1,d+1+n)-d;
dfs(id,0);
int ans=*max_element(d+1,d+1+n)-1;
cout<<ans<<endl;
```

快快编程1961

树上找到三个节点ABC

max{ min{dst(A,B),dst(A,C)} + dst(B,C) }

解法思路的形成
如何启发思路

A 暴力

B 简化为链

C 手算样例

解法思路的形成
如何启发思路

解法1

枚举转折点
预计算前3长路径

解法2

直径BC+枚举A

解法1

枚举转折点
预计算前3长路径

解法1

枚举转折点
预计算前3长路径

```
12  void dfs_h(ll u,ll fa){
13      h[u][1]=h[u][2]=h[u][3]=0;
14      for(ll i=hd[u];i;i=e[i].nxt){
15          ll v=e[i].v,w=e[i].w;
16          if(v==fa)continue;
17          dfs_h(v,u);
18          if(h[u][3]>=h[v][1]+w)continue;
19
20
21
22
23
24      }
25  }
```

解法1

枚举转折点
预计算前3长路径

```
26 void dfs_g(ll u,ll fa){
27     for(ll i=hd[u];i;i=e[i].nxt){
28         ll v=e[i].v,w=e[i].w;
29         if(v==fa)continue;
30         if(                    )
31                                 
32         else
33             g[v]=w+max(g[u],h[u][1]);
34         dfs_g(v,u);
35     }
36 }
```

解法2 直径BC+枚举A

解法2

直径BC+枚举A

```
63      ll ans=0;
64      for(ll A=1;A<=n;++A){
65          ll AB=dst(A,B);
66          ll AC=dst(A,C);
67          ll cost=min(AB,AC)+dst(B,C);
68          ans=max(ans,cost);
69      }
```

快快编程作业

1685

1961

1962