信奥算法

次大值问题

次大值

有n名学生的考试成绩，求排名第1第2的两位学生的分数。

输入样例：
5
0 3 1 2 3

输出样例：
3 3

输入样例：
7
0 3 5 1 2 4 3

输出样例：
5 4

打擂台求最大和次大值

有两个擂主参与守擂

挑战者先和谁打？

次大值

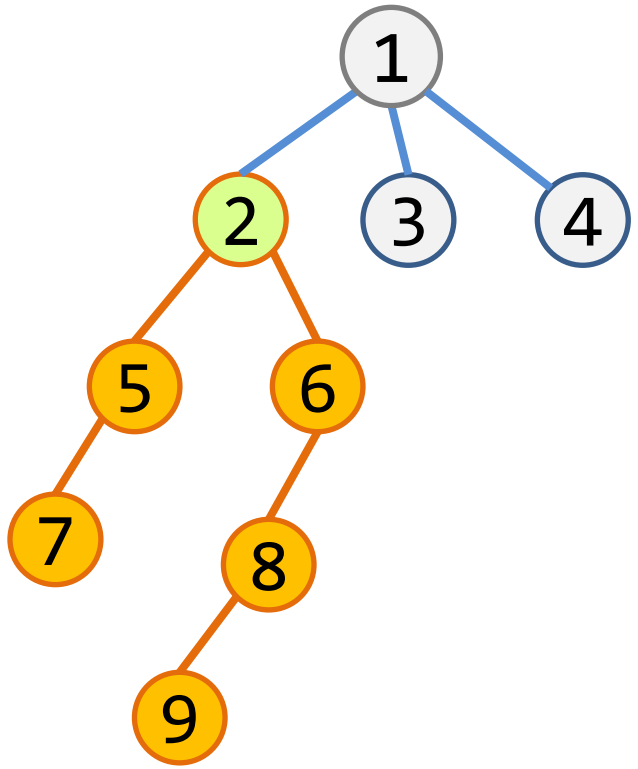
```
 7      cin>>n;
 8      for(int i=1;i<=n;++i)cin>>x[i];
 9      int h1=-INF, h2=-INF;
10      for(int i=1;i<=n;++i){
11          if(x[i]>h2) h2=x[i];
12          if(h2>h1)
13      }
14      cout<<h1<<" "<<h2<<endl;
```

树的直径

diameter

两点间最长的距离

树上任意两点间的 路径唯一确定
路径形态一定是 先往上爬 再往下爬
u,v间路径的转折点 就是LCA(u,v)



直径是5

两点间最长的距离

树的直径

任意两点间最长的距离

算法1

枚举所有点对u,v
求出LCA(u,v)
再求出dst(u,v)

复杂度
不低于$O(n^2)$

算法2

枚举起点u作为根
DFS计算其他节点的深度$d_u[]$
深度减1就是到根的距离

复杂度
$O(n^2)$

算法3

枚举所有转折点/分割点u

在u为根的子树里预计算什么？

以u为起点向下分出
2路最长路径
(路径不可以重叠)

目标复杂度
$O(n)$

h1[u]代表以u为起点向下路径最长长度

h2[u]代表先删除h1[u]路径的边
再计算以u为起点向下路径最长长度
简记为"次长"路径

注意和高度概念有区别

注意h1,h2对应路径不重叠

f[u]代表以u为转折点的路径最长长度

dfs_h1h2()预计算h1,h2数组

```
24 void solve(){
25     dfs_h1h2(1,0);
26     for(int u=1;u<=n;++u) f[u]=
27     int ans=*max_element(f+1,f+1+n);
28     cout<<ans<<endl;
29 }
```

```cpp
void input(){
    cin>>n;
    for(int i=1;i<=n-1;++i){
        int u,v; cin>>u>>v;
        to[u].push_back(v);
        to[v].push_back(u);
    }
}
void solve(){
    dfs_h1h2(1,0);
    for(int u=1;u<=n;++u) f[u]=h1[u]+h2[u];
    int ans=*max_element(f+1,f+1+n);
    cout<<ans<<endl;
}
int main(){
    freopen("diameter.in","r",stdin);
    freopen("diameter.out","w",stdout);
    input();
    solve();
    return 0;
}
```

函数封装
模块化

h1[u]代表以u为起点向下路径最长长度

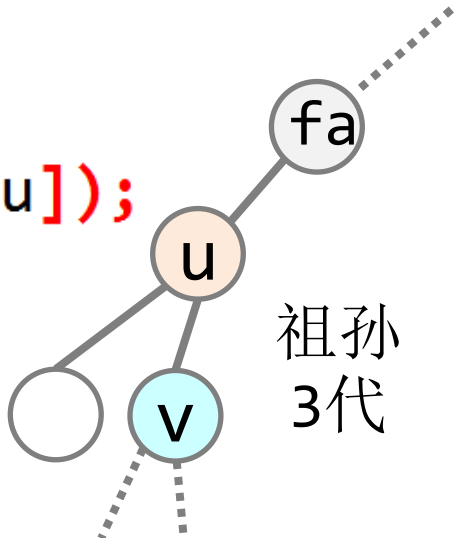
h2[u]代表先删除h1[u]路径的边
再计算以u为起点向下路径最长长度
简记为"次长"路径

注意和高度概念有区别

注意h1,h2对应路径不重叠

```
6  void dfs_h1h2(int u,int fa){
7      h1[u]=h2[u]=0;
8      for(int i=0;i<to[u].size();++i){
9          int v=to[u][i];
10         if(v==fa)continue;
11         dfs_h1h2(v,u);
12         h2[u]=max(h2[u],h1[v]+1);
13         if(h2[u]>h1[u])swap(h2[u],h1[u]);
14     }
15 }
```

易错点:uv混淆12混淆



快快编程870

最长距离

对树上每个点，求其他节点到它的最长距离

算法1

对于每个节点u作为根
DFS计算其他节点的深度$d_u[]$
深度减1就是到根的距离

复杂度
$O(n^2)$

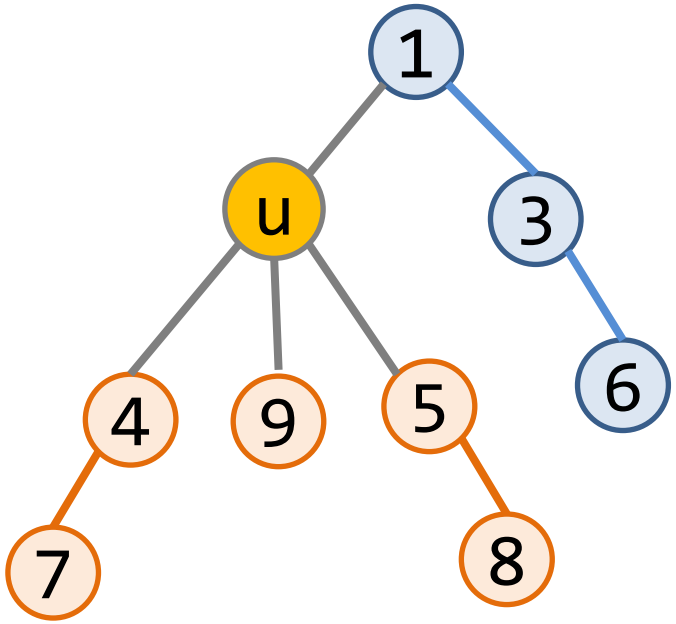
算法2

对于每个节点u作为起点
计算到其他点的最长距离
希望利用一些公共信息
不要每次都重新跑DFS

起点u到其他点的最长距离
依赖哪些基础信息?

目标复杂度
$O(n)$

f[u]代表以u为起点到其他节点的最长距离



计算f[u]需要两种信息

从u起步**往下**走的最长路径	
记作h1[u]	
计算顺序	从下往上

从u起步**往上**走的最长路径	
记作g[u]	
计算顺序	从上往下

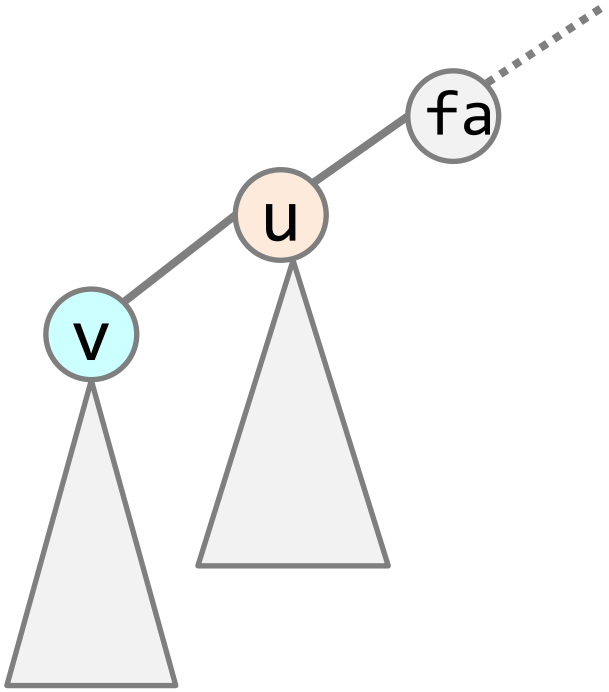
起步往下永远往下

起步往上可改往下

g[u]代表从u起步往上走的最长路径
g[v]代表从v起步往上走的最长路径

计算顺序	从上往下

g[u]已经计算完毕如何推导g[v]

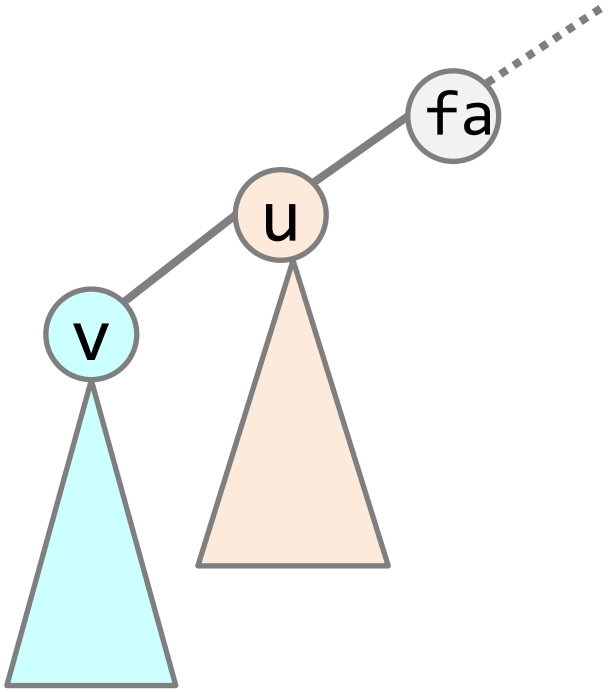


g[v]=g[u]+1	不对！

缺少从v往上走到u 再从u往下走的最长路

从u往下走的最长路 是h1[u]吗？	不对！
从u往下走的最长路 有可能经过v	

g[u]代表从u起步往上走的最长路径
g[v]代表从v起步往上走的最长路径
h1[u]代表从u起步往下走的最长路径
h2[u]代表从u起步往下走的次长路径(与h1[u]不重叠)



如果从u往下走的最长路经过v
if(h1[u]==h1[v]+1)

```
g[v]=1+max(g[u],h2[u]);
```

如果从u往下走的最长路不经过v
if(h1[u]!=h1[v]+1)

```
g[v]=1+max(g[u],h1[u]);
```

h1[u]代表从u起步往下走的最长路径
h2[u]代表从u起步往下走的次长路径(与h1[u]不重叠)

```
 6 void dfs_h1h2(int u,int fa){
 7     for(int i=0;i<to[u].size();++i){
 8         int v=to[u][i];
 9         if(v==fa)continue;
10         dfs_h1h2(v,u);
11         h2[u]=max(h2[u],h1[v]+1);
12
13     }
14 }
```

g[u]代表从u起步往上走的最长路径
g[v]代表从v起步往上走的最长路径
h1[u]代表从u起步往下走的最长路径
h2[u]代表从u起步往下走的次长路径(与h1[u]不重叠)

```
15 void dfs_g(int u,int fa){
16     for(int i=0;i<to[u].size();++i){
17         int v=to[u][i];
18         if(v==fa)continue;
19         if(h1[u]==h1[v]+1)
20             g[v]=
21         else
22             g[v]=
23         dfs_g(v,u);
24     }
25 }
```

计算顺序从上往下
算完g[v]再递归

g[u]代表从u起步往上走的最长路径
g[v]代表从v起步往上走的最长路径
h1[u]代表从u起步往下走的最长路径
h2[u]代表从u起步往下走的次长路径(与h1[u]不重叠)

```
26 void input(){
27     cin>>n;
28     for(int i=1;i<=n-1;++i){
29         int u,v; cin>>u>>v;
30         to[u].push_back(v);
31         to[v].push_back(u);
32     }
33 }
34 void solve(){
35     dfs_h1h2(1,0);
36     dfs_g(1,0);
37     for(int u=1;u<=n;++u)
38         cout<<                    <<endl;
39 }
```

太戈编程

1685

870