

# 树的直径

## Diameter of Tree

**Concept:** 树的直径是指树中最长的一条简单路径的长度

## 两次 dfs

**Theorem:** 从任意一点开始 dfs 一遍，找到距离起点最远的点  $x$ ，则  $x$  一定是直径的一个端点；再从  $x$  开始 dfs 一遍，找到距离  $x$  最远的点  $y$ ，则  $x$  和  $y$  就是树的直径。

**Complexity:**  $O(n)$

**Code:**

```
1  int dis[N];
2  void dfs(int x, int f, int d, int &p){
3      dis[x] = d;
4      if(!p || dis[p] < d)    p = x;
5      for(int i = head[x]; i; i = edge[i].nxt){
6          if(edge[i].to == f) continue;
7          dfs(edge[i].to, x, d + edge[i].dis, p);
8      }
9  }
10
11 int main(){
12     //...
13     int x = 0, y = 0;
14     memset(dis, 0, sizeof dis); dfs(1, 0, 0, x);
15     memset(dis, 0, sizeof dis); dfs(x, 0, 0, y);
16     // now x, y are ends of the diameter
17     printf("%d\n", dis[y]);
18     return 0;
19 }
```

## 树形 dp

**Idea:** 设  $dp[i]$  表示以  $i$  为根的子树中，一个端点在  $i$  的最长路径长度，则在 dp 过程中记录子树的最大、次大长度进行转移和更新答案即可。

**Complexity:**  $O(n)$

**Code:**

```
1  int dp[N], ans;
2  void dfs(int x, int f){
3      int mx1 = 0, mx2 = 0;
4      for(int i = head[x]; i; i = edge[i].nxt){
5          if(edge[i].to == f) continue;
6          dfs(edge[i].to, x);
7          if(mx1 < dp[edge[i].to] + edge[i].dis)
8              mx2 = mx1, mx1 = dp[edge[i].to] + edge[i].dis;
9          else if(mx2 < dp[edge[i].to] + edge[i].dis)
10             mx2 = dp[edge[i].to] + edge[i].dis;
11      }
```

```
12     dp[x] = mx1;
13     ans = max(ans, mx1 + mx2);
14 }
15
16 int main(){
17     //...
18     dfs(1, 0);
19     printf("%d\n", ans);
20     return 0;
21 }
```