# 树的重心

## 定义

如果在树中选择某个节点并删除，这棵树将分为若干棵子树，统计子树节点数并记录最大值。取遍树上所有节点，使此最大值取到最小的节点被称为整个树的重心。

（这里以及下文中的「子树」若无特殊说明都是指无根树的子树，即包括「向上」的那棵子树，并且不包括整棵树自身。）

## 性质

- 树的重心如果不唯一，则至多有两个，且这两个重心相邻。
- 以树的重心为根时，所有子树的大小都不超过整棵树大小的一半。
- 树中所有点到某个点的距离和中，到重心的距离和是最小的；如果有两个重心，那么到它们的距离和一样。
- 把两棵树通过一条边相连得到一棵新的树，那么新的树的重心在连接原来两棵树的重心的路径上。
- 在一棵树上添加或删除一个叶子，那么它的重心最多只移动一条边的距离。

## 求法

根据重心的定义及其第三条性质，有两种方法可以在 $O(n)$ 时间内求出树的所有重心，其中，$n$ 为树的大小。

### DFS 统计子树大小

在 DFS 中计算每个子树的大小，记录「向下」的子树的最大大小，利用总点数减去当前子树（这里的子树指有根树的子树）的大小得到「向上」的子树的大小，然后就可以依据定义找到重心了。

**参考实现**

```cpp
const int MAXN = 50005;

int n;
// 这份代码默认节点编号从 1 开始，即 i ∈ [1,n]
int siz[MAXN],  // 这个节点的「大小」（所有子树上节点数 + 该节点）
    weight[MAXN];  // 这个节点的「重量」，即所有子树「大小」的最大值
vector<int> centroids;  // 用于记录树的重心（存的是节点编号）
vector<int> g[MAXN];

void dfs(int cur, int fa) {  // cur 表示当前节点 (current)
  siz[cur] = 1;
  weight[cur] = 0;
  for (int v : g[cur]) {
    if (v != fa) {  // v 表示这条有向边所通向的节点
      dfs(v, cur);
      siz[cur] += siz[v];
      weight[cur] = max(weight[cur], siz[v]);
    }
  }
  weight[cur] = max(weight[cur], n - siz[cur]);
  if (weight[cur] <= n / 2) {  // 依照树的重心的定义统计
    centroids.push_back(cur);
  }
}

void get_centroids() { dfs(1, 0); }
```

## 换根 DP

根据「树中所有点到某个点的距离和中，到重心的距离和是最小的；如果有两个重心，那么到它们的距离和一样」这一点，我们只需要找出到所有点距离之和最小的点即可。

**参考实现**

```cpp
const int N = 50005;

int n, siz[N];
long long dp[N], ans[N];
vector<int> g[N], centroids;

// 求 1 号节点到所有其他节点的距离和
void dfs1(int u, int fa) {
  siz[u] = 1;
  dp[u] = 0;
  for (int v : g[u]) {
    if (v == fa) continue;
    dfs1(v, u);
    siz[u] += siz[v];
    dp[u] += dp[v] + siz[v];  // 子树节点到 u 的距离和
  }
}

// 通过换根 DP 求所有节点为树根时对应的距离和
void dfs2(int u, int fa) {
  for (int v : g[u]) {
    if (v == fa) continue;
    ans[v] = ans[u] - siz[v] + (n - siz[v]);
    dfs2(v, u);
  }
}

// 求树的重心
void get_centroids() {
  dfs1(1, 0);
  ans[1] = dp[1];
  dfs2(1, 0);

  long long mini = std::numeric_limits<long long>::max();
  for (int i = 1; i <= n; i++) {
    if (ans[i] < mini) {
      mini = ans[i];
      centroids = {i};
    } else if (ans[i] == mini)
      centroids.push_back(i);
  }
}
```

## 例题

### Codeforces Round 359 (Div. 1) B. Kay and Snowflake

给定一棵有根树，求出每一棵子树（有根树意义下且包含整棵树本身）的重心是哪一个节点。

### 解题思路

本题中子树无特殊说明指的是有根树意义下且包含整棵树本身的「向下」的子树。

根据第四条性质，对于一棵以点 $u$ 为根的子树，其重心一定在所有以 $u$ 的直接子节点为根的子树的重心到点 $u$ 的路径上。

类似于上文提到的 DFS 求重心方法，对于每棵以节点 $u$ 为根的子树，先求出所有以其直接子节点为根的子树的重心（叶子节点的重心是其本身），然后向上判断路径上的节点是不是重心即可。

时间复杂度为 $O(n)$ 可以求出所有子树的重心。

参考代码

```cpp
#include <iostream>
#include <vector>
using namespace std;

constexpr int N = 3e5 + 5;

int n, q;  // 点数，询问数
int fa[N];
vector<int> son[N];
int siz[N],     // 子树大小
    ans[N],     // 以节点 u 为根的子树重心是 ans[u]
    weight[N];  // 节点重量

void dfs(int u) {
  siz[u] = 1, ans[u] = u;
  for (int v : son[u]) {
    dfs(v);
    siz[u] += siz[v];
    weight[u] = max(weight[u], siz[v]);
  }
  for (int v : son[u]) {
    int p = ans[v];
    while (p != u) {
      if (max(weight[p], siz[u] - siz[p]) <= siz[u] / 2) {
        ans[u] = p;
        break;
      } else
        p = fa[p];
    }
  }
}

int main() {
  ios::sync_with_stdio(false);
  cin >> n >> q;
  for (int v = 2; v <= n; v++) cin >> fa[v],
son[fa[v]].push_back(v);
  dfs(1);
  while (q--) {
    int u;
    cin >> u;
    cout << ans[u] << '\n';
  }
  return 0;
}
```

习题

- Gym 101649G Godfather
- POJ 1655 Balancing Art
- 洛谷 P1364 医院设置
- Codeforces 1406C Link Cut Centroids
- Codeforces 708C Centroids

## 参考资料


- 树的 "重心" 的一些性质及动态维护 - fanhq666（博客园转载）
- 树的直径、树的重心与树的点分治 - cyendra
- 树的重心的性质及其证明 - suxxsfe
- 《信息学奥林匹克辞典》2.4.7.11 章 1. 树的重心


---