长剖

主要思想：所有非长链的儿子都可以 $O(dep)$ 暴力。

CF1009F

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6;
int n, dfn[N + 5], sign, len[N + 5], son[N + 5], mxv[N + 5], mxd[N + 5], d[N + 5], f[N
+ 5];
vector<int> g[N + 5];
void dfs(int x, int fa) {
    d[x] = d[fa] + 1;
    for (int y : g[x]) {
        if (y == fa) continue;
        dfs(y, x);
        if (len[y] + 1 > len[x]) len[x] = len[y] + 1, son[x] = y;
    }
}
void dfs2(int x, int fa) {
    dfn[x] = ++sign;
    if (son[x]) dfs2(son[x], x);
    for (int y : g[x]) {
        if (y == fa || y == son[x]) continue;
        dfs2(y, x);
    }
}
void Upd(int x, int val, int dep) {
    if (val > mxv[x] || (val == mxv[x] && dep < mxd[x])) mxv[x] = val, mxd[x] = dep;
}
void dfs3(int x, int fa) {
    if (son[x]) {
        dfs3(son[x], x);
        mxv[x] = mxv[son[x]], mxd[x] = mxd[son[x]];
    }
    f[dfn[x]] = 1, Upd(x, 1, d[x]);
    for (int y : g[x]) {
        if (y == fa || y == son[x]) continue;
        dfs3(y, x);
        for (int i = 0; i <= len[y]; i++) {
            f[dfn[x] + i + 1] += f[dfn[y] + i];
            Upd(x, f[dfn[x] + i + 1], d[y] + i);
        }
    }
}
int main() {
    scanf("%d", &n);
    for (int i = 1; i < n; i++) {
```

```
        int x, y;
        scanf("%d%d", &x, &y);
        g[x].push_back(y), g[y].push_back(x);
    }
    dfs(1, 0), dfs2(1, 0), dfs3(1, 0);
    for (int i = 1; i <= n; i++) cout << mxd[i] - d[i] << '\n';
}
```

P5903

求 x 的 k 级祖先。设 $k = 2^t + v\ (v < 2^t)$，先用倍增跳到第 2^t 级祖先 y。注意 y 向下的最长链长度至少是 2^t，也就至少是 v，所以对于每条长链预处理其链长长度的祖先即可。

注意开数组咋开的。

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned int ui;
typedef long long ll;
const int N = 1e6;
ui s;
inline ui get(ui x) {
    x ^= x << 13;
    x ^= x >> 17;
    x ^= x << 5;
    return s = x;
}
int n, q, root;
ll ans = 0;
int sign, len[N + 5], son[N + 5], l2[N + 5] = {-1}, top[N + 5], p[N + 5][21], f[N + 5],
pos[N + 5],
                                                          d[N + 5];
vector<int> g[N + 5];
void DFS1(int x, int fa) {
    d[x] = d[fa] + 1, p[x][0] = fa;
    for (int i = 1; i <= 18; i++) p[x][i] = p[p[x][i - 1]][i - 1];
    for (int y : g[x]) {
        DFS1(y, x);
        if (len[y] + 1 > len[x]) len[x] = len[y] + 1, son[x] = y;
    }
}
void DFS2(int x, int tp) {
    top[x] = tp;
    if (son[x]) DFS2(son[x], tp);
    for (int y : g[x]) {
        if (y == son[x]) continue;
        DFS2(y, y);
    }
}
int Query(int x, int k) {
    if (!k) return x;
    x = p[x][l2[k]], k -= (1 << l2[k]);
    return f[pos[top[x]] + (d[x] - d[top[x]]) - k];
```

```
}
int main() {
    scanf("%d%d%u", &n, &q, &s);
    for (int i = 1, x; i <= n; i++) {
        scanf("%d", &x);
        if (x) g[x].push_back(i);
        else root = i;
    }
    for (int i = 1; i <= n; i++) l2[i] = l2[i / 2] + 1;
    DFS1(root, 0);
    DFS2(root, root);
    int cur = 0;
    for (int i = 1; i <= n; i++) {
        if (i != top[i]) continue;
        pos[i] = cur + len[i] + 1;
        for (int j = len[i], x = i; j >= 0; j--, x = p[x][0]) f[cur + j + 1] = x;
        for (int j = 0, x = i; j <= len[i]; j++, x = son[x]) f[cur + len[i] + 1 + j] =
x;
        cur += len[i] * 2 + 1;
    }
    ui lastans = 0;
    for (int i = 1; i <= q; i++) {
        int x = (get(s) ^ lastans) % n + 1;
        int k = (get(s) ^ lastans) % d[x];
        lastans = Query(x, k);
        ans ^= (1ll * i * lastans);
    }
    printf("%lld\n", ans);
}
```

P4292

看到平均值先二分答案 mid，平均值 $\geq mid$ 等价于所有边权减去 mid 后和 ≥ 0。

然后变成求长度在 $[L,U]$ 之间的路径的最大权值。

设 $f(x,d)$ 表示 x 开始长度为 d 的路径的最大长度。

沿用 CF1009F 的技巧，把 $f(x,d)$ 存在数组里 dfn_x+d 的位置。

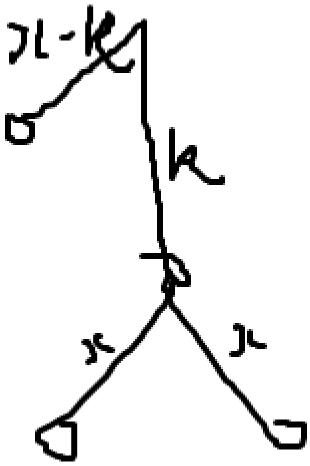
继承长儿子时，把 f 数组区间加，线段树维护。

合并答案时，线段树查询区间最大值。

P5904

考虑满足条件的三个点会呈现什么样子。

会呈现下面的样子：



设 $f(i,x)$ 为 i 的 x 级后代数量，$g(i,y)$ 为 i 开始 (k,x,x) 三条边，$x-k=y$ 的数量。

转移：f 很容易，$g(i,y)=\sum_{u\neq v\in son_i} f(u,y-1)f(v,y-1)+\sum_u g(u,y+1)$

算答案：$\sum_{u\neq v\in son_x} f(u,k)g(v,k)$。

前面可以直接做。后面需要用指针。

指针的方法是：预先分配内存。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int n, m, son[200005], len[200005];
vector<int> gg[200005];
ll *f[200005], *g[200005], F[800005], *nowf, ans;
void DFS1(int x, int fa) {
    len[x] = 1;
    for (int y : gg[x]) {
        if (y == fa) continue;
        DFS1(y, x);
        if (len[y] + 1 > len[x]) len[x] = len[y] + 1, son[x] = y;
    }
}
void assign(int x) { f[x] = nowf, nowf += len[x] << 1, g[x] = nowf, nowf += len[x] <<
1; }
void DFS2(int x, int fa) {
    // cout << x << ' ' << len[x] << '\n';
    if (son[x]) {
        f[son[x]] = f[x] + 1, g[son[x]] = g[x] - 1;
        DFS2(son[x], x);
    }
    f[x][0] = 1, ans += g[x][0]; // 注意这里，统计只有重儿子一个分支的贡献，其它贡献在下面能统计
到!
    for (int y : gg[x]) {
        if (y == fa || y == son[x]) continue;
        assign(y), DFS2(y, x);
```

```
            // cout << x << ' ' << y << '\n';
            for (int i = 0; i < len[y]; i++) {
                if (i) ans += f[x][i - 1] * g[y][i];
                ans += g[x][i + 1] * f[y][i];
            }
            for (int i = 0; i < len[y]; i++) {
                g[x][i + 1] += f[x][i + 1] * f[y][i]; // 注意这里
                f[x][i + 1] += f[y][i];
                if (i) g[x][i - 1] += g[y][i];
            }
        }
}
int main() {
    scanf("%d", &n);
    for (int i = 1, x, y; i < n; i++) scanf("%d%d", &x, &y), gg[x].push_back(y),
gg[y].push_back(x);
    DFS1(1, 0);
    nowf = F, assign(1);
    DFS2(1, 0);
    printf("%lld\n", ans);
}
```

P3899

先分类讨论是上面还是下面，上面简单，下面就是求子树内到 x 距离 $\le k$ 的点的 size 和。

唯一的困难是全局加，这个可以打标记，用到的时候往后推。

记得想清楚“标记”的定义：这里的标记指任意时刻，一个位置的值等于其真实值 + 最开始的标记值。

注意标记的另一种定义（“后缀加标记”），如果只有前缀访问，也可以。

LOJ6712

设 $f(i,j)$ 表示 i 子树内，与 i 最远距离为 j，的连通块数。$f(x,i)f(y,j) \to f'(x, \max(i,j+1))$。但是要求 $i+j+1 \le K$。

用线段树维护 $f(i,j)$。

注意 $\to$ 的意思是 `+=`，所以一个不会错的方法（防止 f', f 互相影响）是把所有要进行的修改存下来，先存下来所有修改，再执行修改。

- $i \le j+1$：$f(x,i)f(y,j) \to f'(x,j+1)$，这部分枚举 j，对 i 求区间和（注意加上 K 的限制还是区间和），然后对 f' 单点加。
- $i > j+1$：$f(x,i)f(y,j) \to f'(x,i)$。
    - 对于 $i \le len_y+1$，可以枚举 i，对 j 求区间和（注意加上 K 的限制）。
    - 对于 $i > len_y+1$，如果没有 K 的限制就是区间乘（注意这时，区间加区间乘操作互不干扰的）

      有 K 的限制，就还要求 $j \le K-i-1$。这会影响到的只有 $[K-len_y-1, K-1]$ 区间内的 i，这样的 i 只有 $O(len_y)$ 个，也可以对这部分暴力单点修改。

算答案直接对所有 f 求和。

线段树可以用 `dfn` 的技巧，只用开一棵。

几个练习题：

[FJOI2014] 最短路径树问题，UOJ33, UOJ284