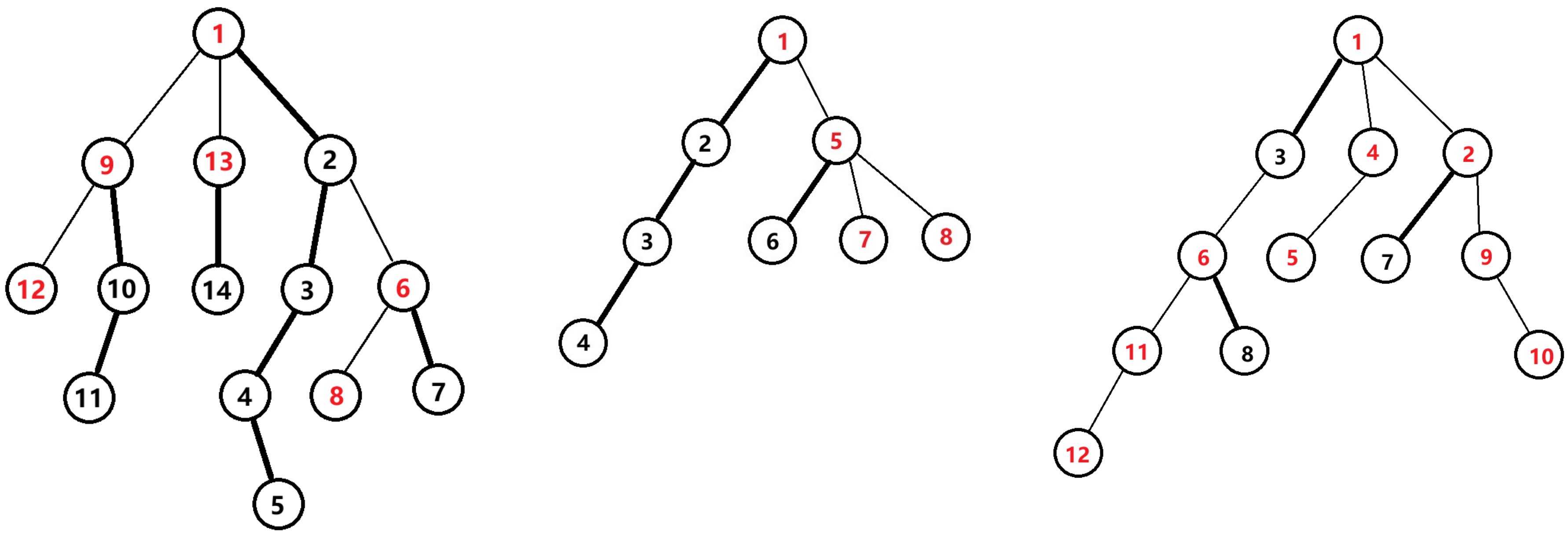
树链剖分

jiangly

什么是**树链剖分**？**树链剖分**有什么用？

- **树链剖分**，顾名思义，就是将**树**剖分成若干条**链**。
  - 这里的**链**，指的是有根树中祖先到后代（直上直下）的路径。
  - 每个点恰好属于一条路径。
  - 对于每个非叶结点，最多只有一个子结点和它属于同一条链。
- 最常见的树链剖分有重链剖分和长链剖分。
- 最常见的作用：
  - 把树上问题转化成我们熟悉的序列问题。
  - 求解一些树上查询，如 LCA、k 级祖先等。
  - 对树进行分治、优化树上 DP 等。

树链剖分的例子



重链剖分

- 重链剖分（Heavy-Light Decomposition，简称 HLD)，也叫轻重链剖分。通常说的树链剖分基本都指重链剖分。
- 重链剖分最重要的用途是得到一个特殊的 DFS 序，使得每条树上路径都能被划分为 $O(\log n)$ 个 DFS 序上的区间，转化为序列上的问题。同时因为是 DFS 序，能兼容子树相关的问题。
- 除此之外，重链剖分也常被用来求 LCA、k 级祖先等。
- 重链剖分还能被扩展为树上启发式合并、静态链分治等。

重链剖分

- 记每个点的子树大小为 $siz(u)$、深度为 $dep(u)$、父节点为 $parent(u)$。
- 重链剖分就是每个点以 $siz(v)$ 最大的子节点 v 作为重子节点 $heavy(u)$。其余子节点称为轻子节点。
- 记每个点所在链的顶端为 $top(u)$、其子树所在的 DFS 区间为 $\left[in(u), out(u)\right)$。
- $seq(i)$ 为 DFS 序中第 i 个点（也就是 $in(u) = i$ 的 u）。

代码实现

```
void dfs1(int u) {
    if (parent[u] ≠ -1) {
        adj[u].erase(std::find(adj[u].begin(), adj[u].end(), parent[u]));
    }

    siz[u] = 1;
    for (auto &v : adj[u]) {
        parent[v] = u;
        dep[v] = dep[u] + 1;
        dfs1(v);
        siz[u] += siz[v];
        if (siz[v] > siz[adj[u][0]]) {
            std::swap(v, adj[u][0]);
        }
    }
}
void dfs2(int u) {
```

代码实现

```cpp
void dfs2(int u) {
    in[u] = cur++;
    seq[in[u]] = u;
    for (auto v : adj[u]) {
        top[v] = v == adj[u][0] ? top[u] : v;
        dfs2(v);
    }
    out[u] = cur;
}
```

例题：最近公共祖先

- 在 $O(n)$ 预处理、$O(\log n)$ 单次询问下解决最近公共祖先 LCA。

```cpp
int lca(int u, int v) {
    while (top[u] != top[v]) {
        if (dep[top[u]] > dep[top[v]]) {
            u = parent[top[u]];
        } else {
            v = parent[top[v]];
        }
    }
    return dep[u] < dep[v] ? u : v;
}
```

例题：k 级祖先

- 在 $O(n)$ 预处理、$O(\log n)$ 单次询问下解决 k 级祖先（也就是一个点往上跳 k 次到达的点）。

```
int jump(int u, int k) {
    if (dep[u] < k) {
        return -1;
    }

    int d = dep[u] - k;

    while (dep[top[u]] > d) {
        u = parent[top[u]];
    }

    return seq[in[u] - dep[u] + d];
}
```

例题：ZJOI 2008 树的统计

- 一棵 n 个点的树，每个点有权值，有三种操作共 q 次：
- CHANGE u t：把 u 的权值改为 t。
- QMAX u v：询问 u 到 v 的路径上结点的权值最大值。
- QSUM u v：询问 u 到 v 的路径上结点的权值和。

- $1 \leq n \leq 30000$，$0 \leq q \leq 200000$。

例题：ZJOI 2008 树的统计

- 树链剖分+线段树。

```
while (top[u] ≠ top[v]) {
    if (dep[top[u]] > dep[top[v]]) {
        在线段树上对区间 [in[top[u]], in[u]] 进行操作;
        u = parent[top[u]];
    } else {
        在线段树上对区间 [in[top[v]], in[v]] 进行操作;
        v = parent[top[v]];
    }
}
if (dep[top[u]] > dep[top[v]]) {
    std::swap(u, v);
}
在线段树上对区间 [in[u], in[v]] 进行操作;
```

- 时间复杂度 $O(n + q \log^2 n)$。
- 注意这道题可以同时加上子树操作。

例题：LNOI 2014 LCA

- 一棵 n 个点的有根树，m 次询问，每个询问给出 l,r,z，求

$$\sum_{i=l}^{r} dep\big(lca(i,z)\big)$$

- 其中根的深度视为 1。
- $1 \le n,m \le 50000$。

例题：LNOI 2014 LCA

- 询问可以差分为 $[1, l-1]$ 和 $[1, r]$。
- 注意到 $dep\big(lca(i,j)\big)$ 相当于 i 和 j 到根的路径上的公共点数。
- 对 i 到根的路径 $+1$，求 j 到根的路径的和，就是答案。
- 枚举 $i = 1, \ldots, n$，对 i 到根的路径 $+1$ 并处理询问。
- 时间复杂度 $O\big((n+m)\log^2 n\big)$。
- 因为 $dis(i,j) = dep(i) + dep(j) - 2dep\big(lca(i,j)\big)$，这个技巧可以被用来维护点集内两两的距离和。

例题：HDOJ 4718 The LCIS on the Tree

- 一棵 n 个点的树，q 次询问，每次给定两个点 u, v，询问 u 到 v 路径上按顺序的点权序列的最长连续上升序列的长度。
- 例如 [1,3,2,4,6,5] 的最长连续上升序列是 [2,4,6]，长度为 3。
- $1 \le n, q \le 100000$。

例题：HDOJ 4718 The LCIS on the Tree

- 只需考虑所有极长的上升段。
- 线段树维护答案、最长上升前缀、最长上升后缀和两端的值（还有整个区间上升的情况）。
- 注意因为路径会是一段往上一段往下、还需要维护下降的答案（一般情况下也就是区间翻转后的答案）。
- 同时在查询的时候要注意合并的顺序。
- 时间复杂度 $O(n + q\log^2 n)$。

例题：CF600E Lomsat gelral

- 一棵 n 个点的有根树，每个点有颜色，对每个子树，求其中出现次数最多的颜色，有多个则求其总和。
- $1 \leq n \leq 10^5$。

例题：CF600E Lomsat gelral

- 树上启发式合并/DSU on tree。
- DSU 指的是不交集合合并，也就是不同子树的合并。
- 可以使用 set、map、哈希表等数据结构来进行启发式合并，但是通过本质相同的做法，可以去掉数据结构，减小常数和实现难度。
- 对每个点 u，首先分别统计其所有轻子树的答案并清空统计值，然后统计其重子树的答案但不清空统计值，接着继续统计其所有轻子树和 u 自己，就得到了 u 子树的答案。
- 时间复杂度 $O(n \log n)$。

例题：CF704E Iron Man

- 一棵 n 个点的树，边长为 1。
- m 个人，第 i 个人从 t_i 时刻开始从 u_i 出发往 v_i 走，速度为 c_i。
- 移动是连续的，一个人可能处于边上的任意一个位置。
- 可能 $u_i = v_i$，代表这个人只在 t_i 时刻出现就立即消失。
- 求两个人碰面的最早时间，如果没有就输出 -1。
- $1 \le n, m \le 100000$。

例题：CF704E Iron Man

- 如果是链上，那么每个人可以看成二维平面的一条线段，一维是时间，一维是位置。
- 只需扫描线就能求出最早的交点。
- 在树上则利用树链剖分把路径转化为 $O(\log n)$ 条链，然后对每条重链分别求解。
- 注意这里的链包含顶端的边。
- 时间复杂度 $O(n + m \log m \log n)$。

例题：ABC311Ex Many Illumination Plans

- 一棵 N 个点的有根树，每个点有美丽度 B、重量 W、颜色 C（0 或 1），对每个子树求以下问题的答案：
  - 你可以重复以下操作任意次：选择一个除根以外的点 u，把 u 的孩子都连到 u 的父亲上，然后删除 u。
  - 最后得到的树应当满足：相邻的点颜色不同、总重量不超过 X。
  - 在上述条件下最大化总美丽度。
- 所有子树的问题是独立的，并不会真的删除点。
- $2 \le N \le 200$，$0 \le X \le 50000$。

例题：ABC311Ex Many Illumination Plans

- 问题转化为选择若干个点，根必须选，并且选的每个点与它上面第一个点颜色不同。
- 直接对子树 DP 的复杂度是 $O(NX^2)$，无法避免合并背包。
- 先考虑只算整棵树的答案。考虑令 $dfs(u,dp)$ 表示 u 的子树在 dp 的基础上算出的 DP 数组。
- 先合并 u 的子树和 dp，再考虑 u 的贡献。
- 如果 u 是叶子，那子树合并后就是 $[dp,dp]$；否则，首先调用 $dfs(heavy(u),dp)$ 得到 f，然后对每棵轻子树分别调用 $dfs(v,f[0])$ 和 $dfs(v,f[1])$ 得到合并后新的 f。

例题：ABC311Ex Many Illumination Plans

- 对每个点，假设它到根的路径上有 d 条轻边，则它对时间复杂度的贡献就是 2^d。
- 要计算每个点的答案时，因为重子树的答案可以继承，不用重新算，所以轻边的贡献变成 $2^d+2^{d-1}+\cdots+1=2^{d+1}-1$，仅仅差了常数。
- 而因为 $d\le\log n$，所以 $2^d\le n$。
- 然而 2^d 的总和并不是 n^2。令这个总和为 $f(u)$，则 $f(u)=1+f\big(heavy(u)\big)+2\sum f(v)\le n^{\log_2 3}$。
- 总复杂度为 $O\big(N^{\log_2 3}X\big)$。