树上倍增

2020年1月22日

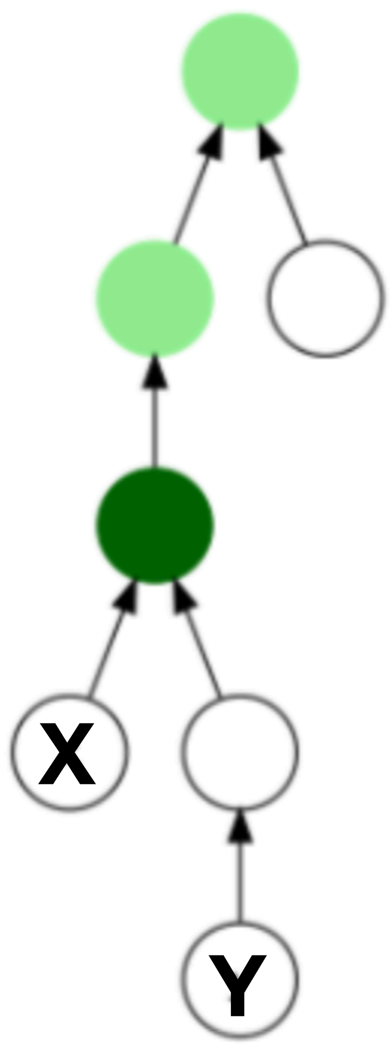
黄哲威 hzwer
北京大学16级计算机科学

课程安排 5

最近公共祖先

树上倍增

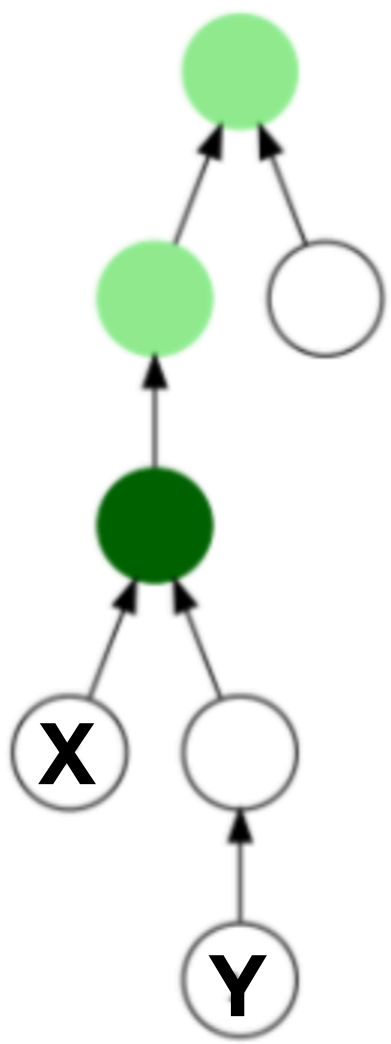
最近公共祖先



在一棵树上，两个结点的最近公共祖先 (LCA) 是它们的公共祖先中深度最大的那个顶点。

比如说，在左边的树中，顶点 x 和 y 的公共祖先用绿色标出，其中深绿色的顶点就是它们的最近公共祖先。

最近公共祖先



求解 LCA 的朴素做法（一）

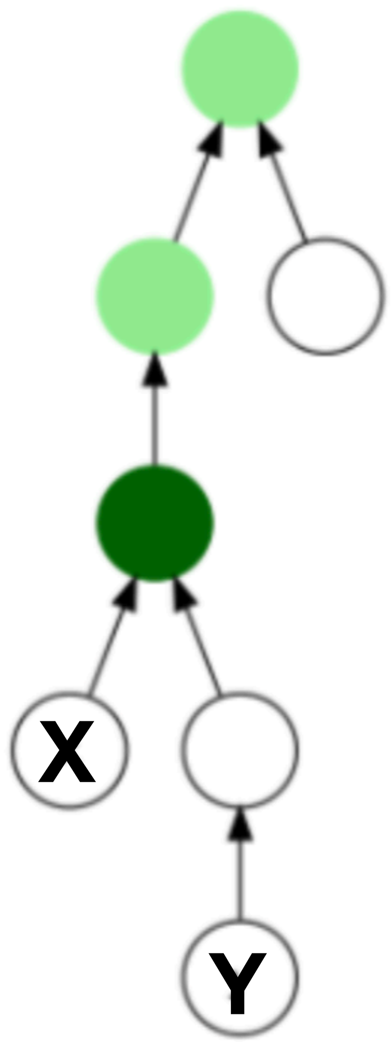
从 X 开始，把它到根的路径都打上标记

从 Y 开始，往上走，找第一个有标记的结点，就是 X 和 Y 的 LCA

最近公共祖先

```
int fa[N];
bool vis[N];
int lca(int x, int y)
{
    memset(vis, 0, sizeof(vis));
    while(x)
    {
        vis[x] = 1;
        x = fa[x];
    }
    while(!vis[y])
        y = fa[y];
    return y;
}
```

最近公共祖先



求解 LCA 的朴素做法（二）

先用 dfs，预处理出每个结点的深度

比较 X 和 Y 的深度，深度较深的那个结点往上跳，深度相同时 X 往上跳

直到 X 和 Y 在 LCA 相遇

最近公共祖先

```cpp
int fa[N], dep[N];
vector<int> e[N];
void dfs(int x) // 根的dep是1
{
    for(int i = 0; i < e[x].size(); i++)
        if(!dep[e[x][i]])
        {
            dep[e[x][i]] = dep[x] + 1;
            dfs(e[x][i]);
        }
}
int lca(int x, int y)
{
    while(x != y)
    {
        if(dep[x] > dep[y])
            x = fa[x];
        else y = fa[y];
    }
    return x;
}
```

快速幂

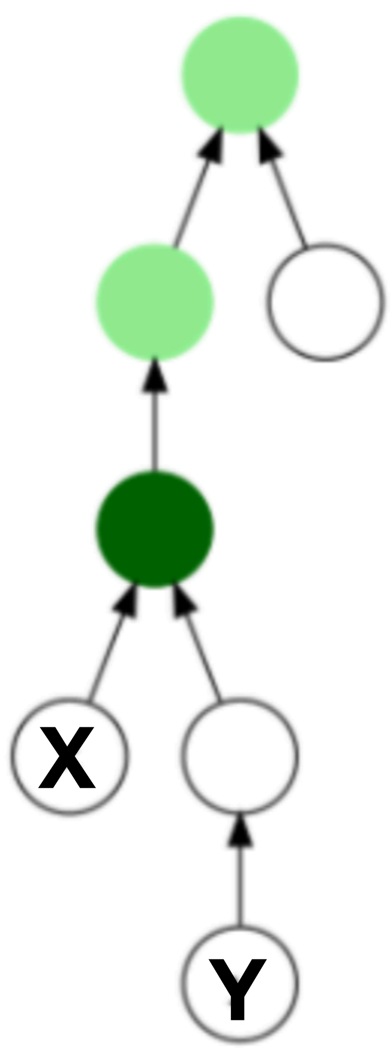
为了理解树上倍增思想。我们先复习一下快速幂。

快速幂 a^b

预处理 a^1 a^2 a^4 … a^(2n)，对 b 做二进制拆分

如 a^21 = a^16 * a^4 * a^1

最近公共祖先



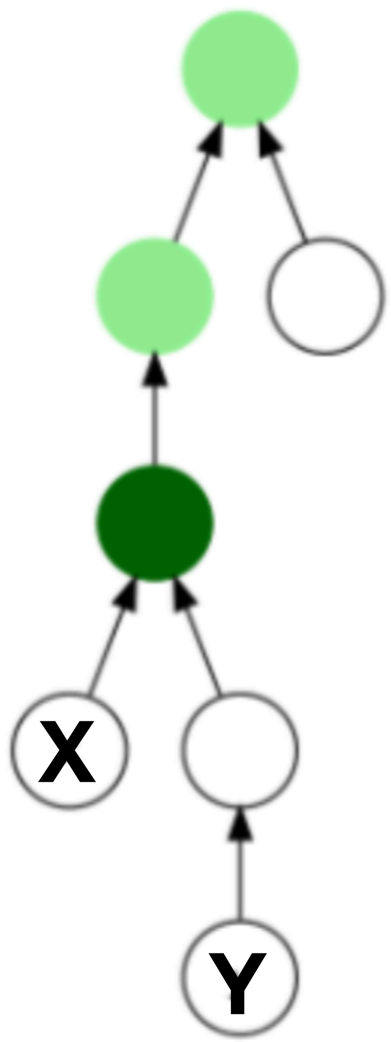
常见的求最近公共祖先的算法是倍增算法。

首先对于每个结点先进行 DFS 预处理出它的深度，再记录下它 们往父亲方向走 2^0, 2^1, 2^2, · · · , 2^k 步所到达的结点。在这里 2^k 大于整棵树的最大深度。

预处理完后，需要查询两个点 u 和 v 的 LCA 的时候，先将 u 和 v 中深度较大的一个利用先前处理出的数组走到和另一个结点 相同的深度，这的所需要的操作次数不会超过

log2 |depth(u) – depth(v)|。

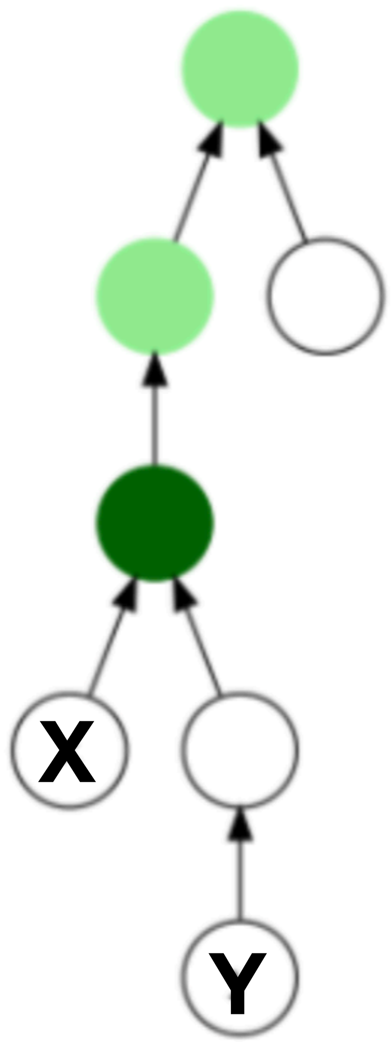
最近公共祖先



接下来从 k 开始往下枚举，如果 u 和 v 如果往上走 2^i 后不 相同，那么就将它们一起往上走这么多步。

结束后如果 u 和 v 仍然不相等，再往上走一步。最后的顶点 就是它们的 LCA。

最近公共祖先



倍增算法预处理的复杂度预处理是 O(nlogn)，每次查询都是 O(logn)。

不仅如此，我们可以动态地给树增加一些叶子结点。

此外，在预处理的时候还可以记录下这段路径的权值最大值，最小值或者权值和之类的信息，这样就可以在 O(logn) 的时间内求出树上两点间路径权值的最大值、最小值还有权值和。

最近公共祖先 (POJ1330)

```
int T, n;
int fa[10005][16], dep[10005], d[10005];
vector<int> e[10005];
void dfs(int u)
{
    for(int i = 1; i <= 15; i++)
        fa[u][i] = fa[fa[u][i - 1]][i - 1];
    for(int i = 0; i < e[u].size(); i++)
    {
        int v = e[u][i];
        dep[v] = dep[u] + 1;
        fa[v][0] = u;
        dfs(v);
    }
}
int lca(int u, int v)
{
    if(dep[u] > dep[v])swap(u, v);
    int t = dep[v] - dep[u];
    for(int i = 15; i >= 0; i--)
        if((1 << i) & t)
            v = fa[v][i];
    if(u == v)return u;
    for(int i = 15; i >= 0; i--)
        if(fa[u][i] != fa[v][i])
            u = fa[u][i], v = fa[v][i];
    return fa[u][0];
}
```

```
int main()
{
    scanf("%d", &T);
    while(T--)
    {
        memset(dep, 0, sizeof(dep));
        memset(fa, 0, sizeof(fa));
        memset(d, 0 ,sizeof(d));
        scanf("%d", &n);
        int u, v;
        for(int i = 1; i < n; i++)
        {
            scanf("%d%d", &u, &v);
            e[u].push_back(v);
            d[v]++;
        }
        scanf("%d%d", &u, &v);
        int rt = 0;
        for(int i = 1; i <= n; i++)
            if(!d[i])rt = i;
        dfs(rt);
        cout << lca(u, v) << endl;
        for(int i = 1; i <= n; i++)
            e[i].clear();
    }
    return 0;
}
```

树上区间 gcd

给一棵 n 个结点的树，第 i 个结点有点权 vi。m 次询问，每次询问 a 到 b 的路径上所有点权的最大公约数。

$n <= 10^5, vi <= 10^9$

树上区间 gcd

gcd 也可以支持区间合并，所以直接用树上倍增来维护。

唯一最小生成树

给出一个有 n 个结点，m 条边的无向图，判断其最小生成树是否唯一。

其中 $0 < n < 10^4, 0 < m < 10^6$。

唯一最小生成树

MST 什么时候唯一?

加入一条非树边会形成环

找到环上除了新加入的边外权值最大的边, 该边权值小于这条非树边

AHOI2008. 紧急集合

给出一个有 n 个结点的树，有 q 次询问，每次询问给三个点 A, B, C，要求找出一个集合地点 O，使得三个点到 O 的距离和最小。

n, q <= 500000

AHOI2008. 紧急集合

对于两个点，最优解一是他们俩的 lca

对于三个点，两两求 lca，可以证明最优解一定在这三个 lca 之中

NOIP2013. 货车运输

n 座城市，编号从 1 到 n，城市之间有 m 条双向道路。 每一条道路对车辆都有限重。现在有 q 辆货车运输货物，每辆货车需要从一个城市运输到另一个城市。司机们想知道每辆车在不超过车辆限重的情况下，最多能运多重的货物。

其中 $n <= 10^4$, $m <= 5×10^4$, $q <= 3×10^4$

NOIP2013. 货车运输

解法1：路径一定在最大生成树上，其实就是求树上两点路径的最小值

解法2：在做最大生成树时，用按秩合并的并查集，在两个点之间暴力找路径

CF519E. A and B and Lecture Rooms

给出一个有 n 个结点的树，有 m 个询问，每次询问一个点对 (a, b)，求有多少个结点 r，满足 r 到 a 和 b 的距离相等。

n, m <= 10^5

CF519E. A and B and Lecture Rooms

如果 a = b，答案是 n。

如果 a, b 深度相等，答案是 lca 为根的树大小，减去 a, b 所在的与 lca 连接的子树大小。

如果 a，b 深度不同，不妨设 a 深于 b，找他们俩路径上的中点 mid，答案是 mid 为根的树大小，减去 a 所在的与 mid 连接的子树大小。

BZOJ2144. 跳跳棋

数轴的整点上有 3 个无差别的棋子，分别在位置 a, b, c。

每次可以选一个棋子，相对另一个棋子进行对称跳动，跳动后它们俩的距离保持不变，棋子始终不能重合。

要通过最少的跳动次数，使得最后棋子位置是 x, y, z。

问最少跳动次数。

$a, b, c <= 10^9$

BZOJ2144. 跳跳棋

考虑三个棋子，两边的棋子有一个能往中间跳，中间的棋子能往两边跳，所有局面状态构成一个二叉树，树的深度不超过 10^9。

若$a < b < c$，记 $b - a$ 为 $t1$，$c - b$ 为 $t2$，不妨设 $t1 < t2$

则处于最左边的棋子最多往中间跳 $(t2 - 1) / t1$ 次，跳的过程其实就是一个辗转相除的过程，因此可以算出每个结点的深度。

可以在虚拟的树上，用二分或者倍增的方法求两个状态的 lca，并且计算路径长度。

CF1229B. Kamil and Making a Stream

给一棵 n 个结点的树，第 i 个结点有点权 vi。对于所有满足条件的路径 (u, v)，其中 u 是 v 的祖先，求路径的最大公约数，再把所有答案求和后输出。

n <= 10^5, vi <= 10^9

CF1229B. Kamil and Making a Stream

做法1：gcd 性质，从一个结点往上走，区间 gcd 最多变化 log 次。

可以用树上倍增求区间 gcd，每次往上走的时候，二分 + 倍增，求出 gcd 变化的边界。

做法2：用 vector 维护一个点往上的不同 gcd，以及它们贡献答案的次数，这个 vector 大小是 log 的。dfs 时暴力往儿子转移。