



TÉLÉCOM PARIS X

TÉLÉCODE

SÉANCE 2 | TÉLÉCODE

AN CÊTRE COMMUN LE PLUS PROCHE (LCA)

Présenté et rédigé par Léopold Bernard

PLAN

01

ARBRES

02

PRÉSENTATION DU PROBLÈME

03

ALGORITHME NAÏF

04

OPTIMISATION

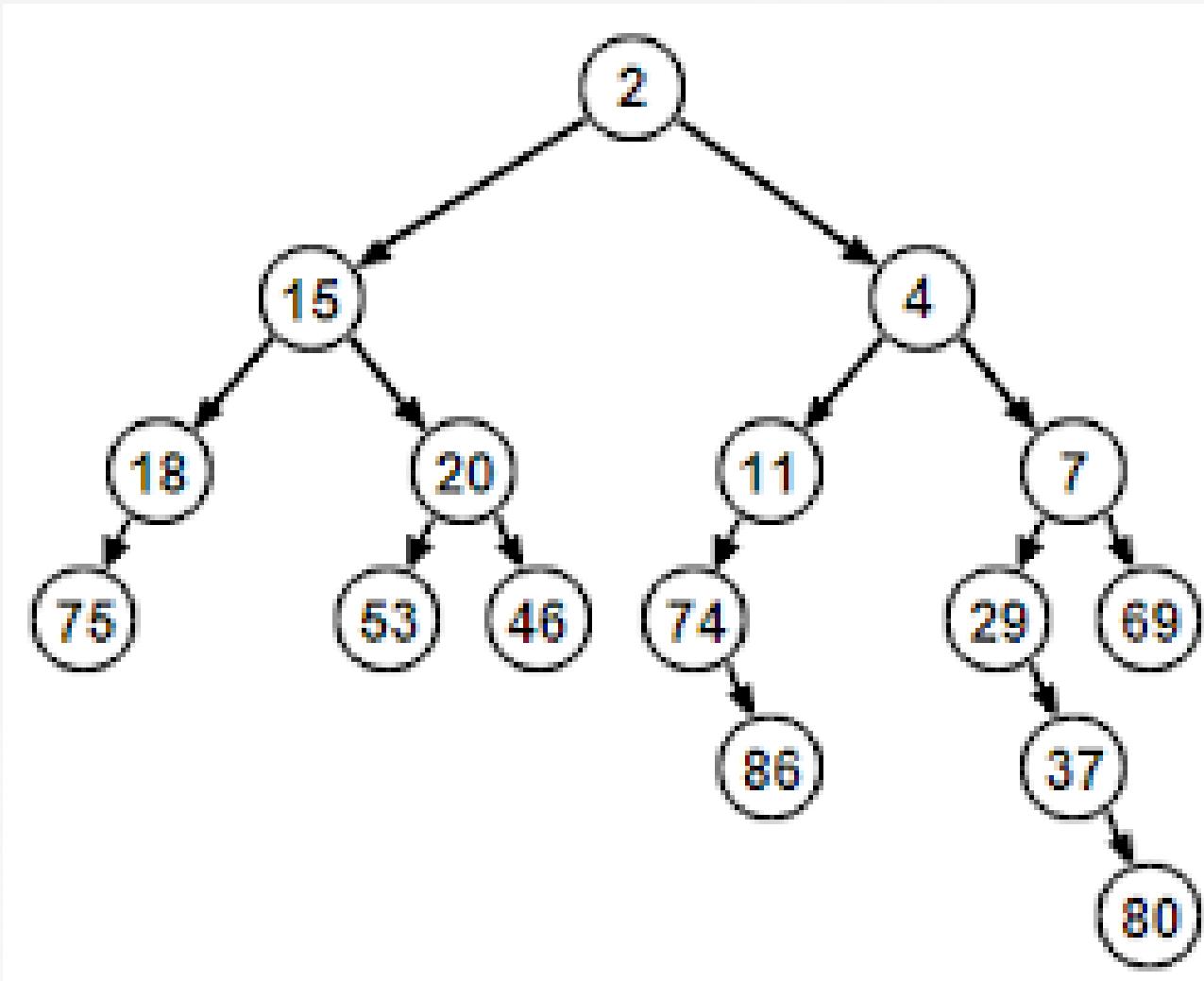
05

ALGORITHME EN $O(\log n)$

06

PROBLÈMES

1. ARBRES



- graphe acyclique orienté
- en général, il est **enraciné**, et sa racine est unique
- chaque nœud possède un unique **parent** (sauf racine)
- **feuille** : nœud ne possédant pas de fils
- **nœud interne** : tout sauf les feuilles

2. PRÉSENTATION DU PROBLÈME

- Ancêtre commun le plus proche = LCA
(least common ancestor)
- Arbre avec n noeuds
- m requêtes de la forme (u, v)
- $LCA(u, v) =$ ancêtre commun le plus proche des nœuds u et v



3. ALGORITHME NAÏF

- On précalcule les hauteurs des noeuds en $O(n)$ avec un DFS
- Pour chaque requête (u, v) :
 - tant que $h(u) < h(v)$ ou $h(u) > h(v)$ on remonte le plus bas de u ou v
 - si $u = v$, on renvoie u
 - sinon, $u = \text{parent}[u]$, $v = \text{parent}[v]$ et on continue



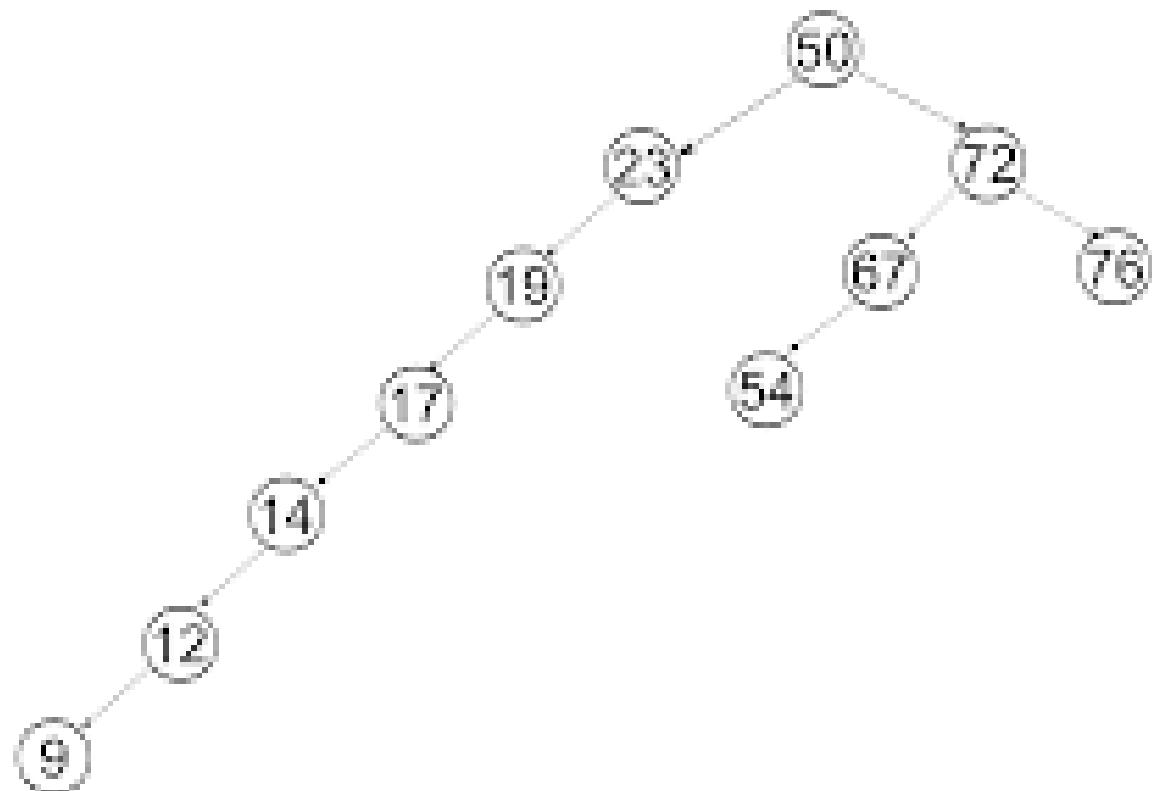
3. ALGORITHME NAÏF

Cas problématiques :

- arbres très “verticaux”

Ici LCA(50, 9) donne ~6 itérations

Pire cas : $O(n)$ itérations



4. OPTIMISATION

- Idée : on va faire des “sauts” de plus en plus grands
- Pour chaque noeud u :
 - $up[u]$ contient ses prédécesseurs à hauteurs $1, 2, 4, \dots, 2^M$ avec $M = \text{ceil}(\log_2(n))$
 - $up[u][k] = (2^k)$ ième parent de u
- On voit arriver une complexité $O(\log N)$...



4. OPTIMISATION

- 2 fonctions :
 - **DFS** : permet de calculer le tableau up efficacement en parcourant le graphe en $O(n \log n)$
 - **is_ancestor(u,v)** : détermine si u est un ancêtre de v en $O(1)$ grâce au précalculs du DFS et les marquages de temps d'entrée (tin) et de sortie (tout)



5. ALGORITHME EN $O(\log N)$

```
void dfs(int v, int p)
{
    tin[v] = ++timer;
    up[v][0] = p;
    for (int i = 1; i <= l; ++i)
        up[v][i] = up[up[v][i-1]][i-1];

    for (int u : adj[v]) {
        if (u != p)
            dfs(u, v);
    }

    tout[v] = ++timer;
}
```

5. ALGORITHME EN $O(\log N)$

```
bool is_ancestor(int u, int v)
{
    return tin[u] <= tin[v] && tout[u] >= tout[v];
}

int lca(int u, int v)
{
    if (is_ancestor(u, v))
        return u;
    if (is_ancestor(v, u))
        return v;
    for (int i = 1; i >= 0; --i) {
        if (!is_ancestor(up[u][i], v))
            u = up[u][i];
    }
    return up[u][0];
}
```

6. PROBLÈMES

Leetcode :

1483. Kth ancestor of a tree node

Codeforces :

Satyam and counting : 2009/D

Ticket hoarding : 1951/C

Maximize the root : 1997/D