

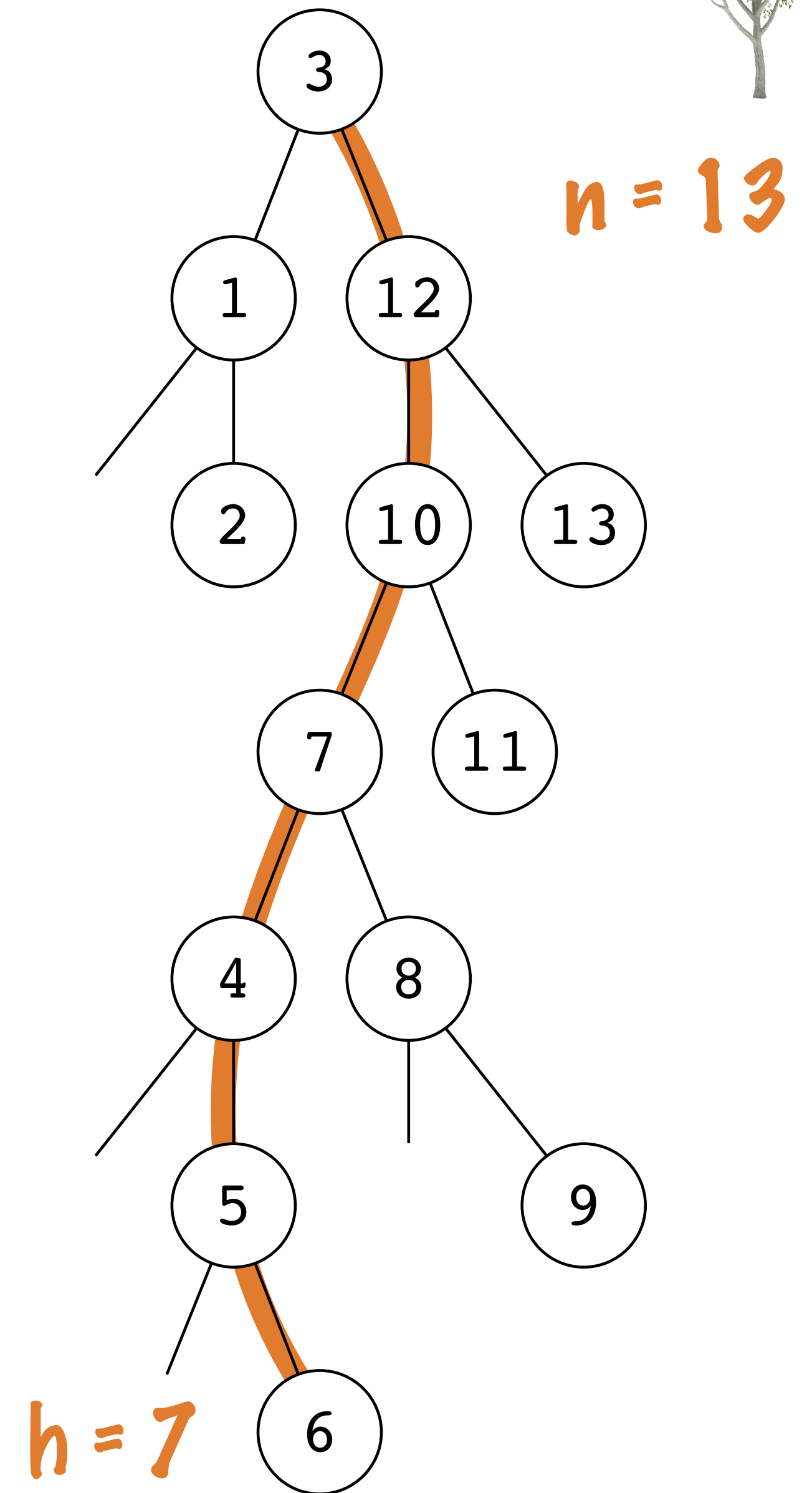
13 . Complexité des opérations



Complexités pour un arbre de taille n et hauteur h



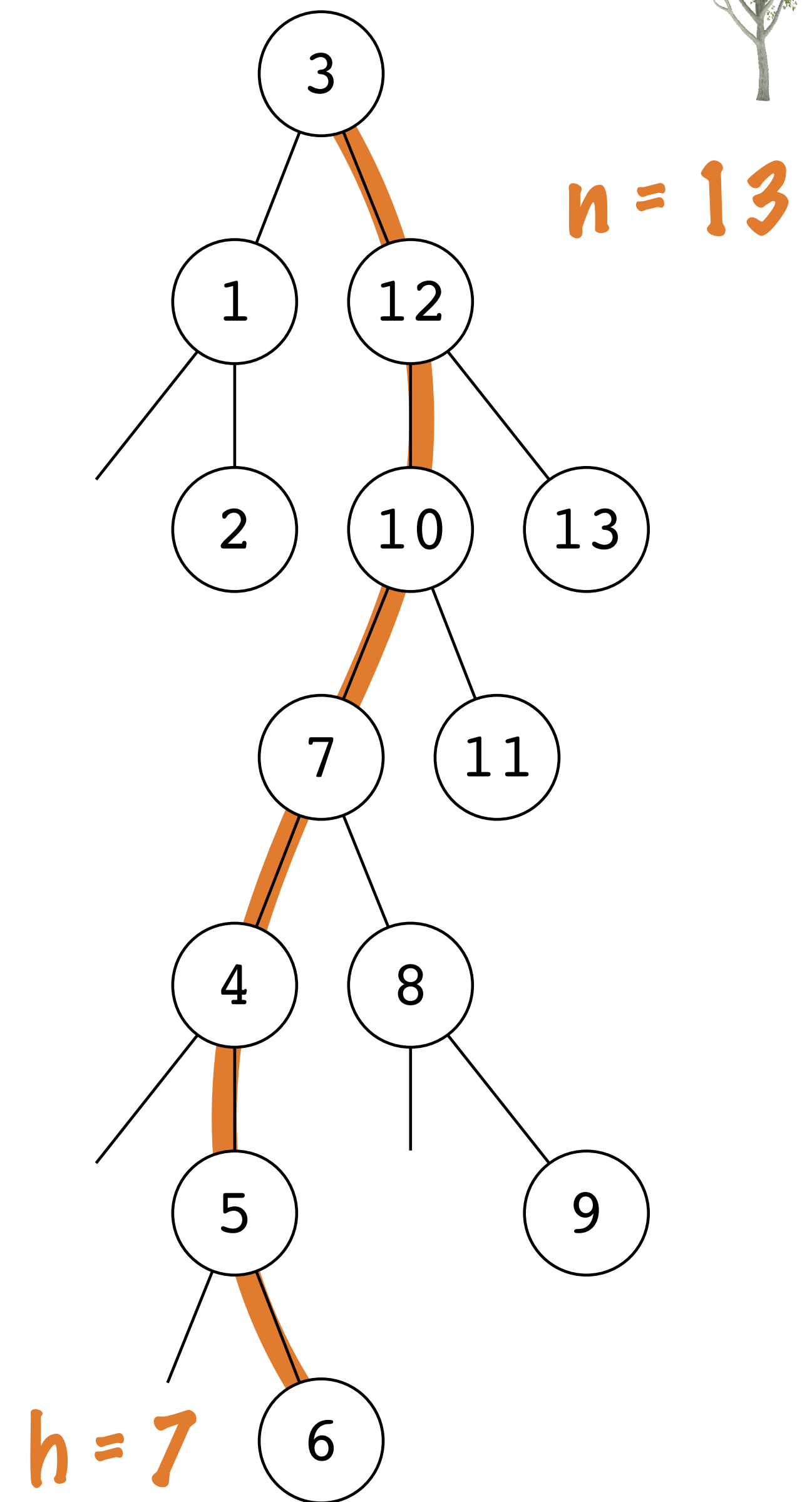
- Parcours en



Complexités pour un arbre de taille n et hauteur h



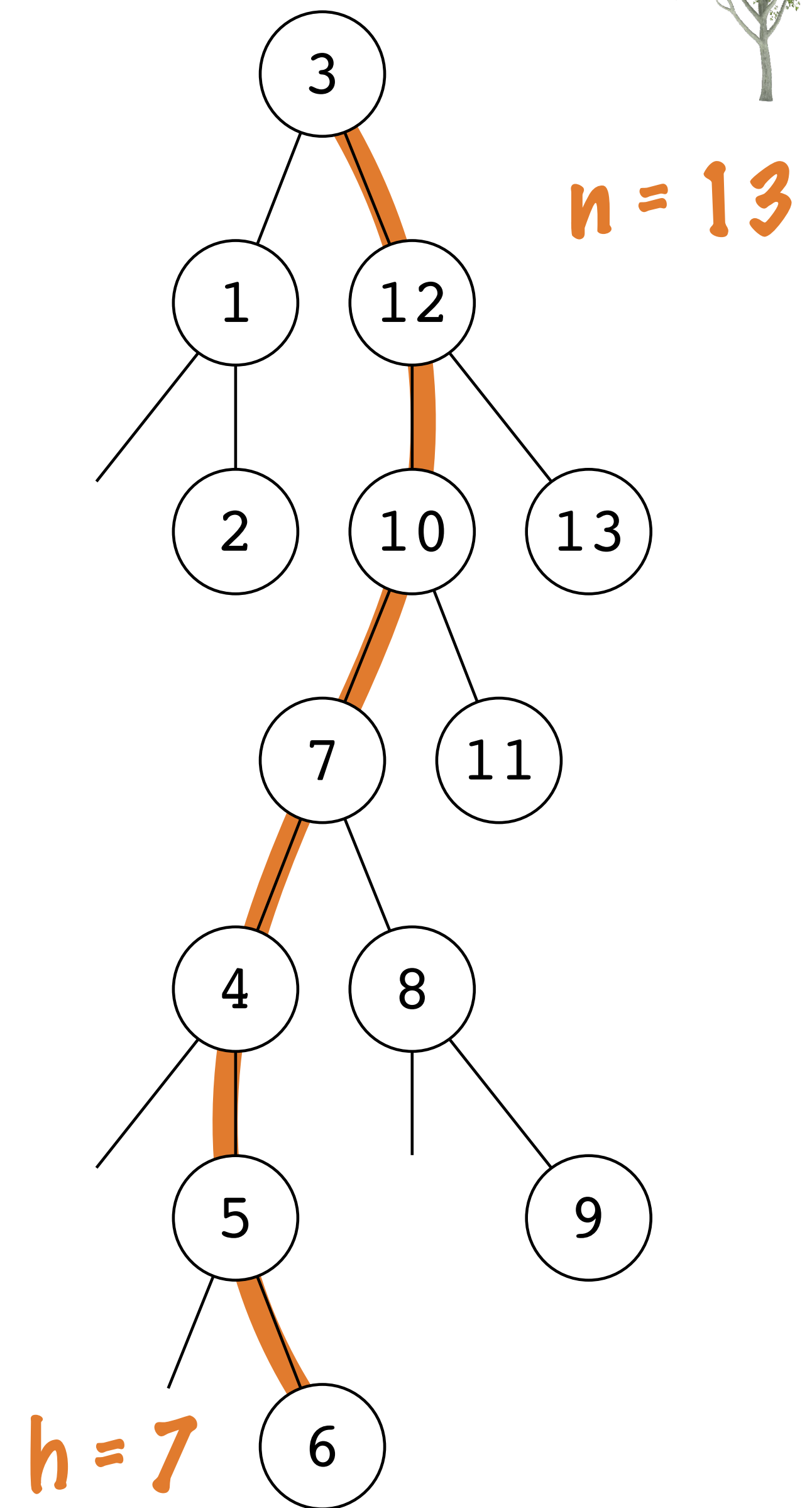
- Parcours en $O(n)$
- Itération (avec lien parent)
 - sur tout l'arbre en



Complexités pour un arbre de taille n et hauteur h



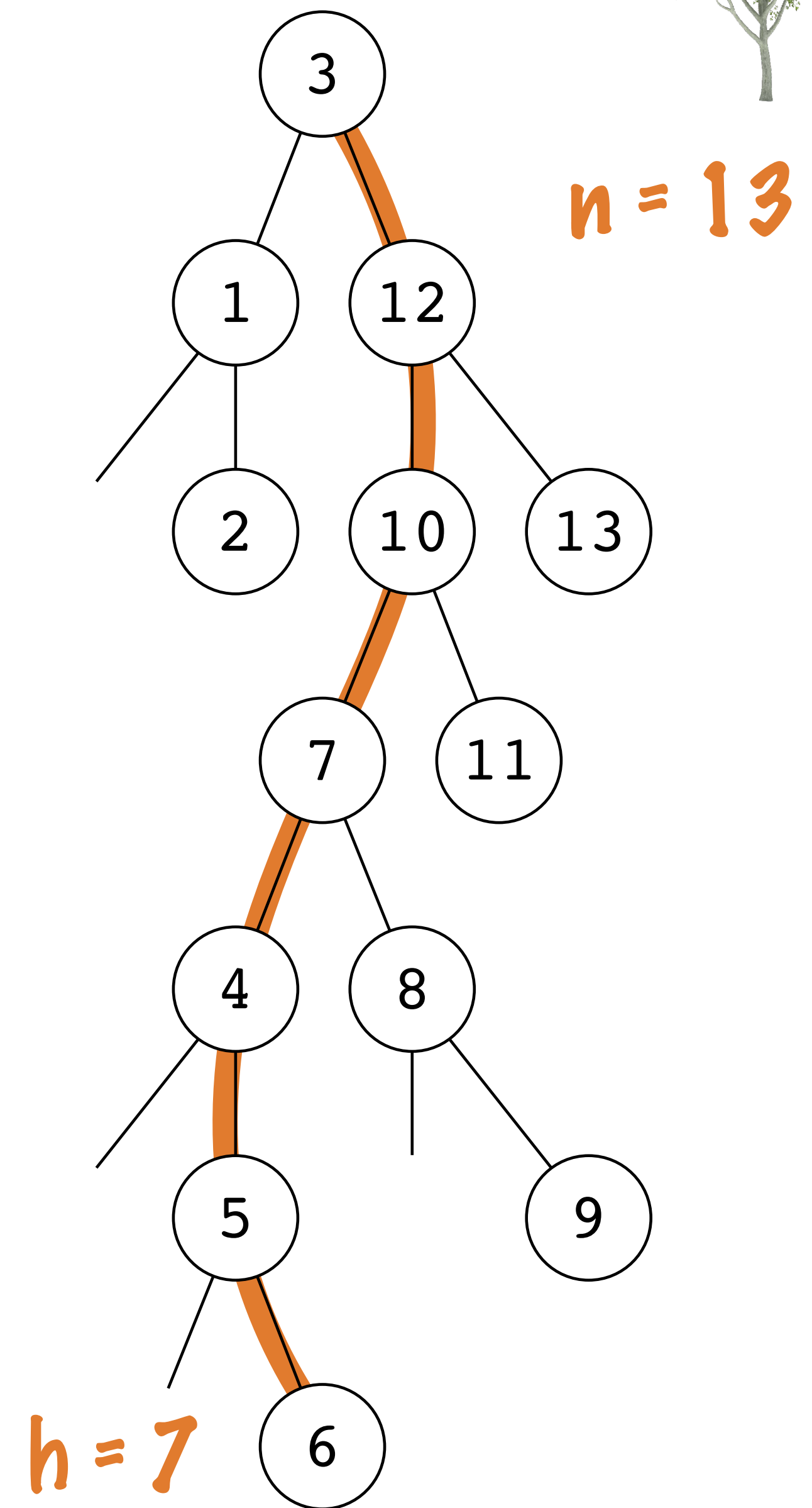
- Parcours en $O(n)$
- Itération (avec lien parent)
 - sur tout l'arbre en $O(n)$
 - suivant en



Complexités pour un arbre de taille n et hauteur h



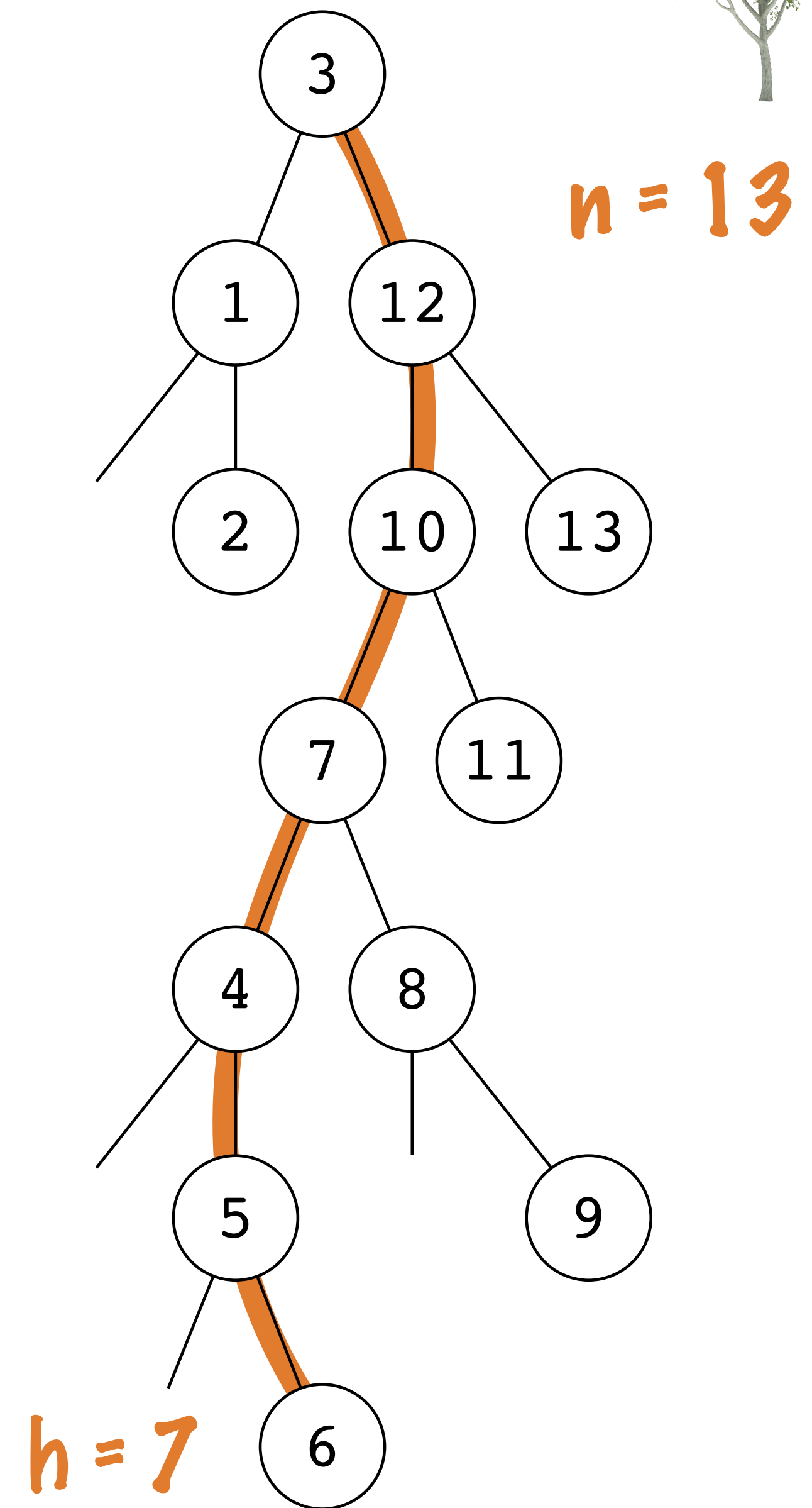
- Parcours en $O(n)$
- Itération (avec lien parent)
 - sur tout l'arbre en $O(n)$
 - suivant en $O(1)$ en moyenne, $O(h)$ au pire.



Complexités pour un arbre de taille n et hauteur h



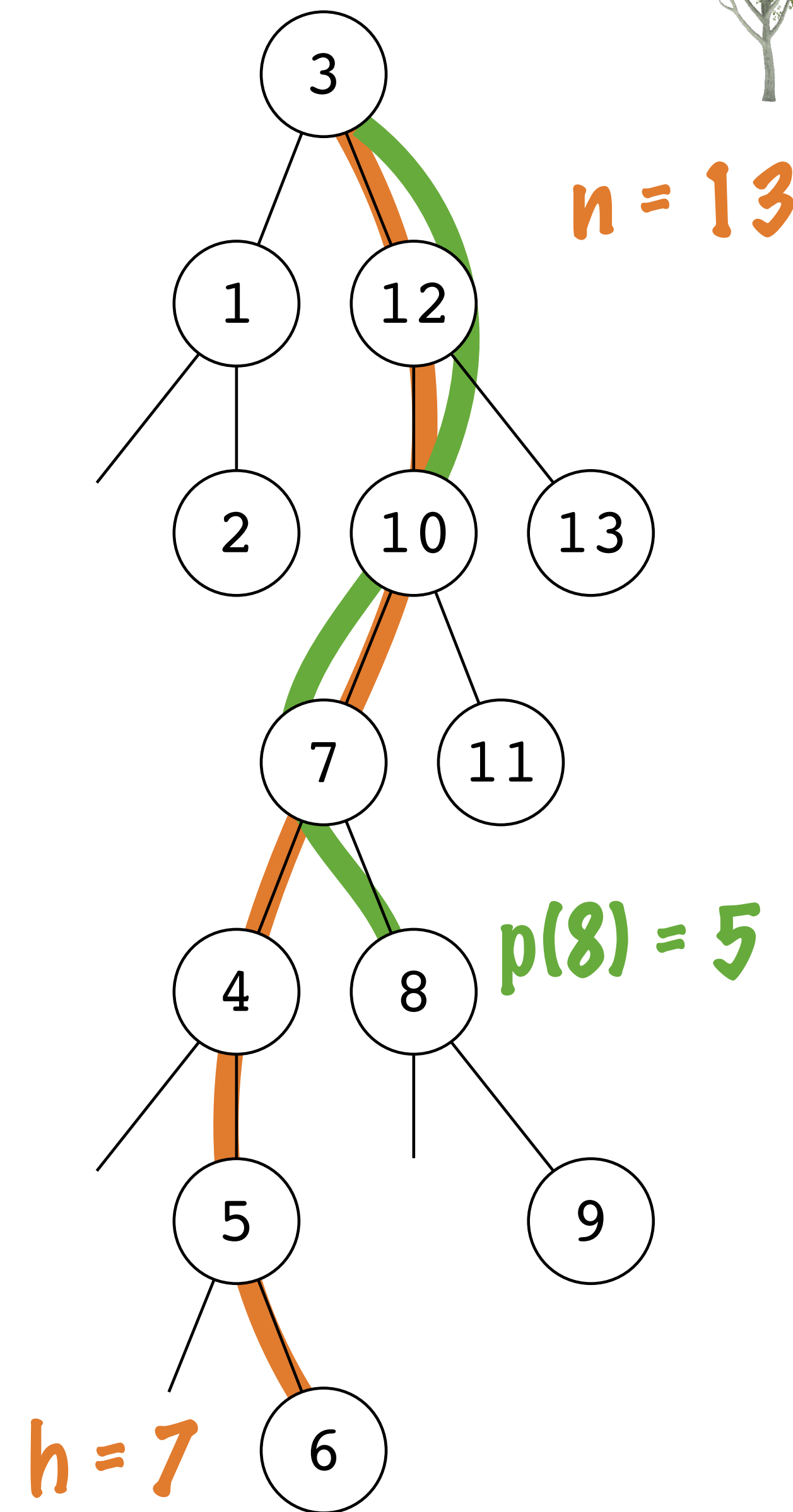
- Parcours en $O(n)$
- Itération (avec lien parent)
 - sur tout l'arbre en $O(n)$
 - suivant en $O(1)$ en moyenne, $O(h)$ au pire.
- Recherche, insertion, suppression, min et max en



Complexités pour un arbre de taille n et hauteur h



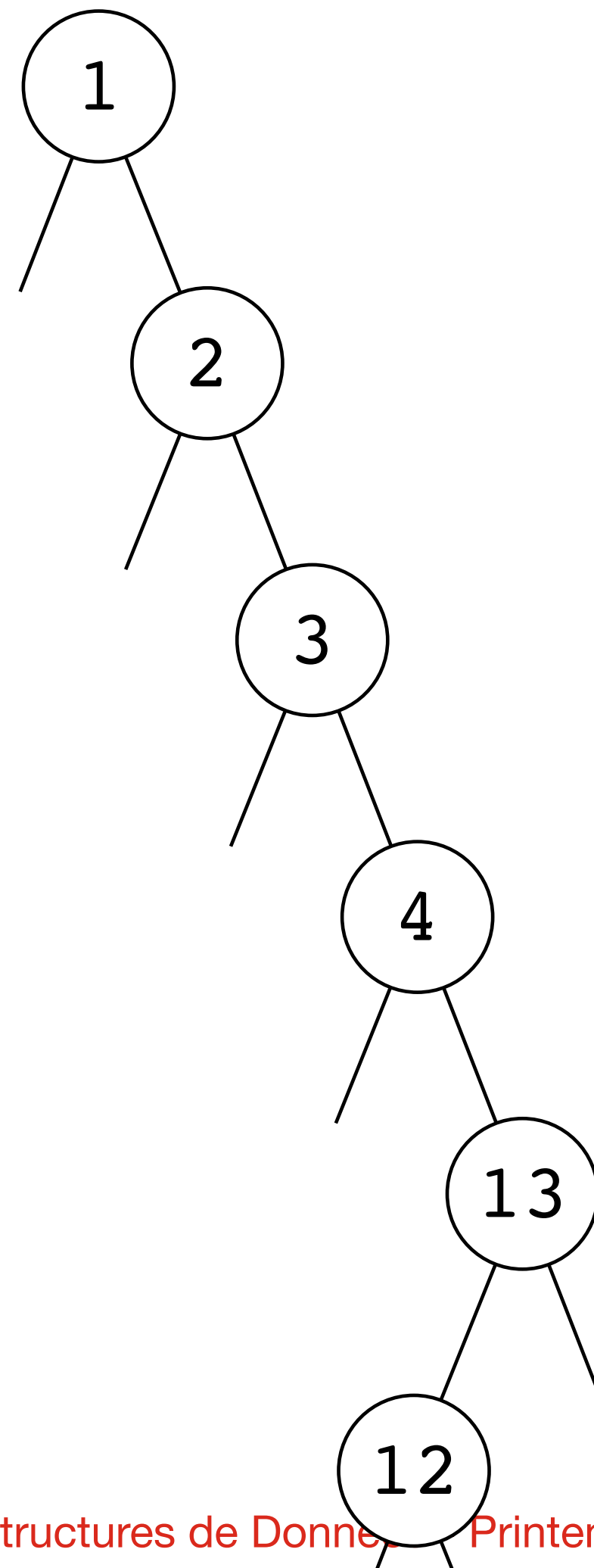
- Parcours en $O(n)$
- Itération (avec lien parent)
 - sur tout l'arbre en $O(n)$
 - suivant en $O(1)$ en moyenne, $O(h)$ au pire.
- Recherche, insertion, suppression, min et max en $O(p)$ avec p le niveau du noeud concerné
- Quelle sont les relations entre n , h et p ?



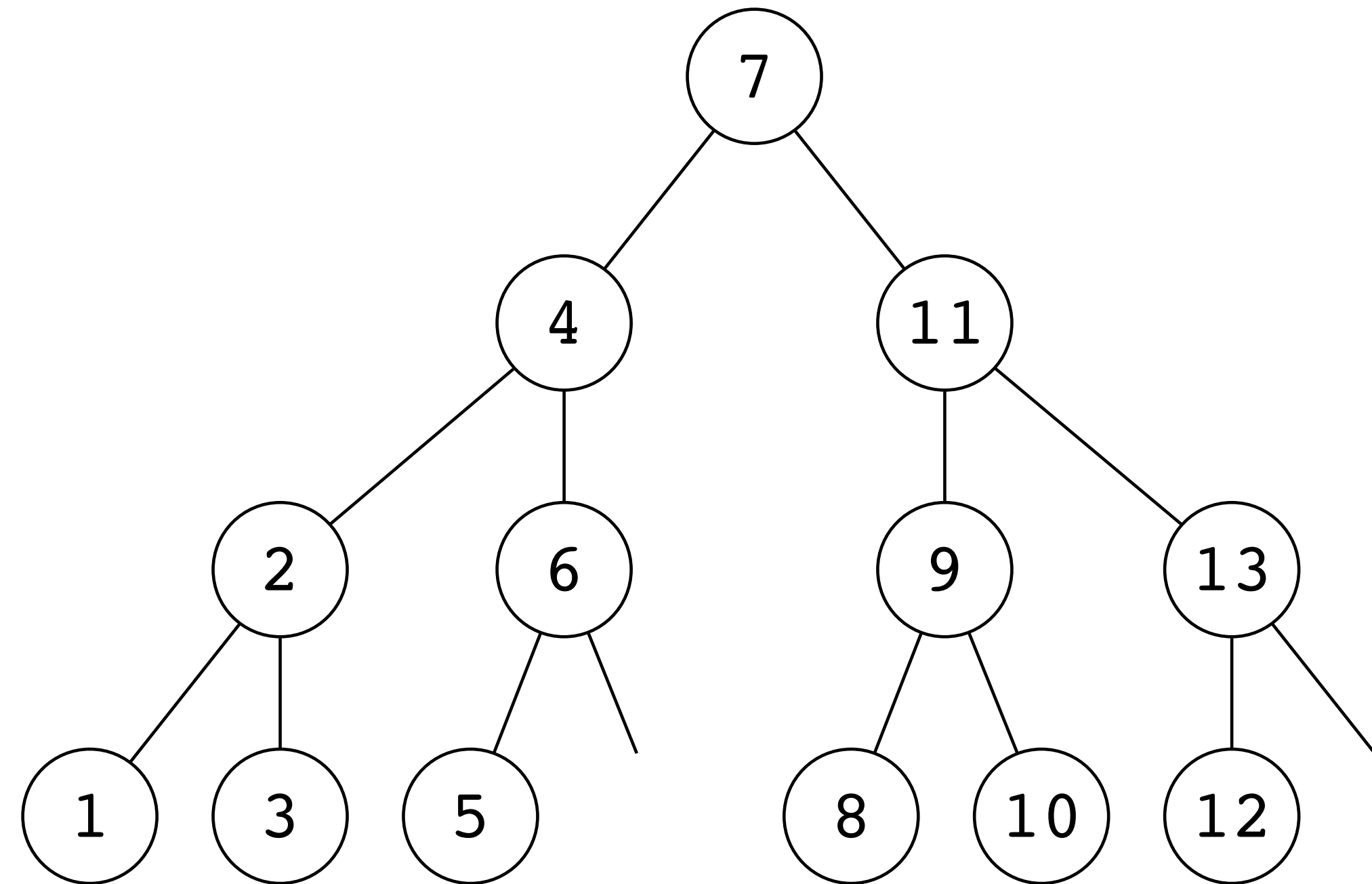
n, h et p



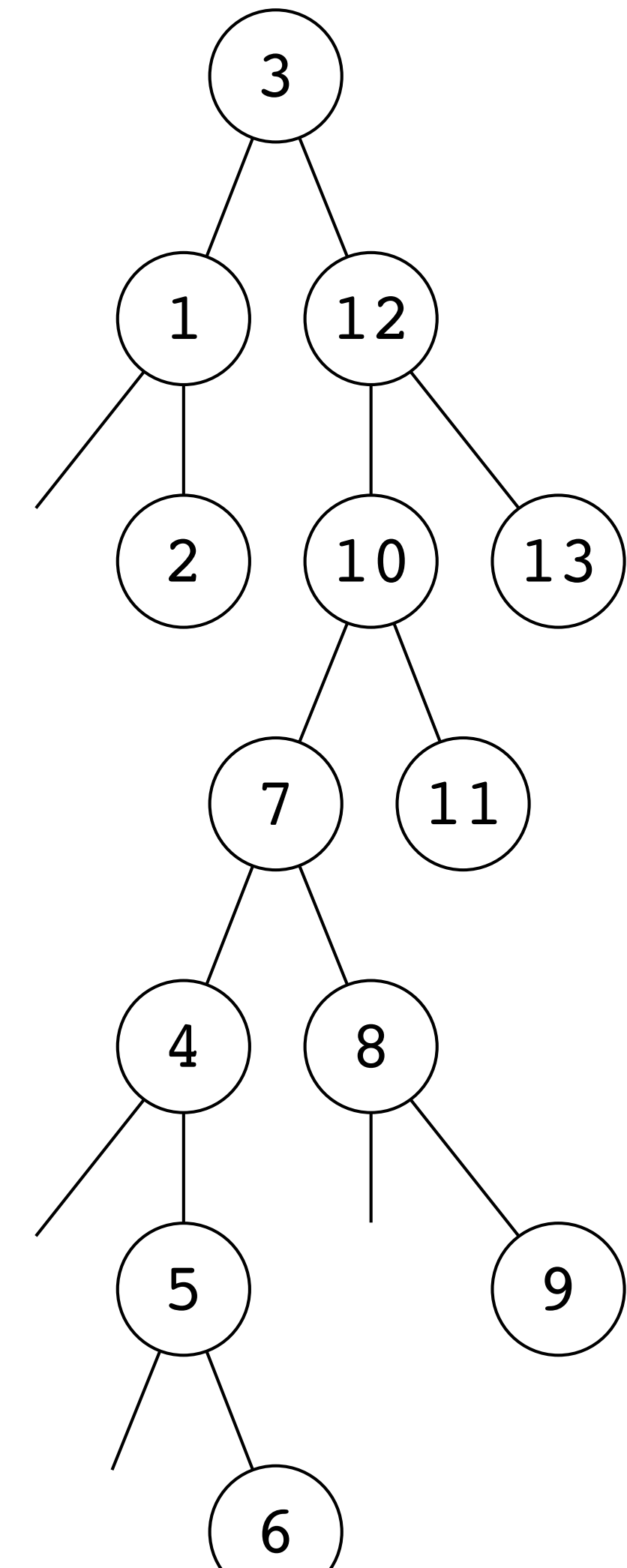
Dégénéré
(Pire cas)



Plein
(Meilleur cas)



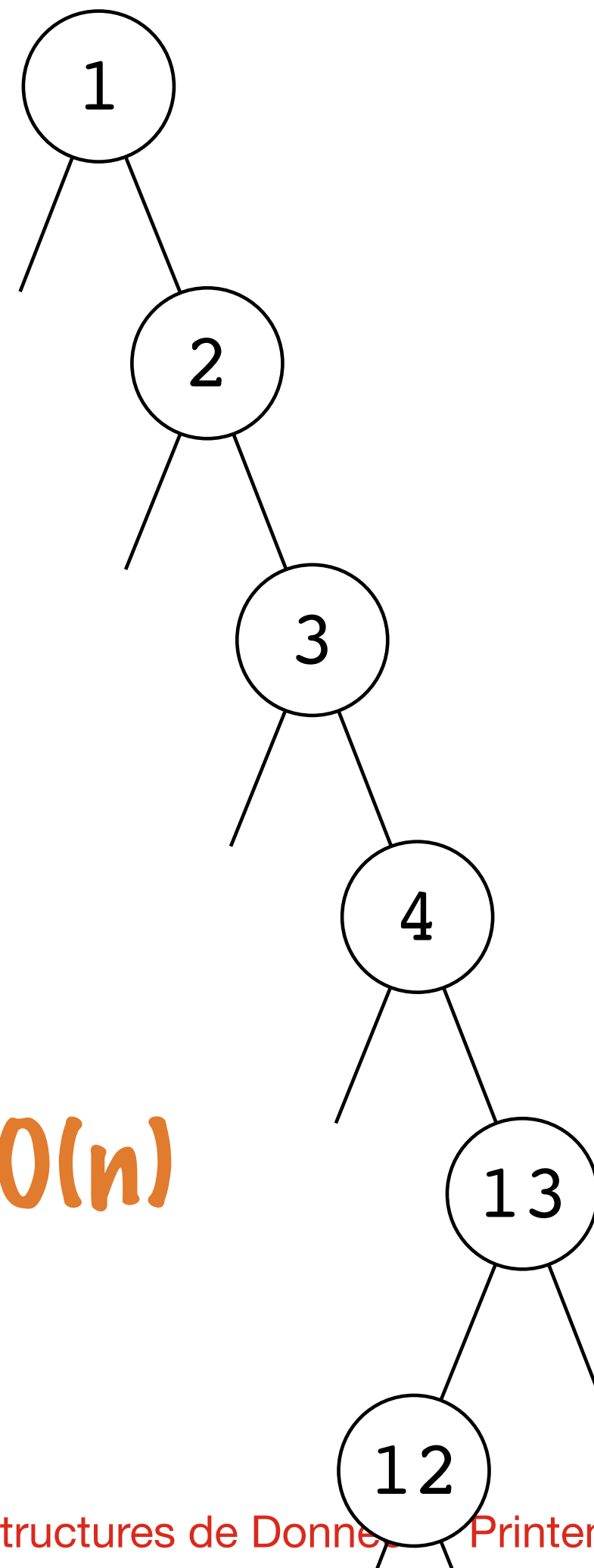
Quelconque
(Aléatoire)





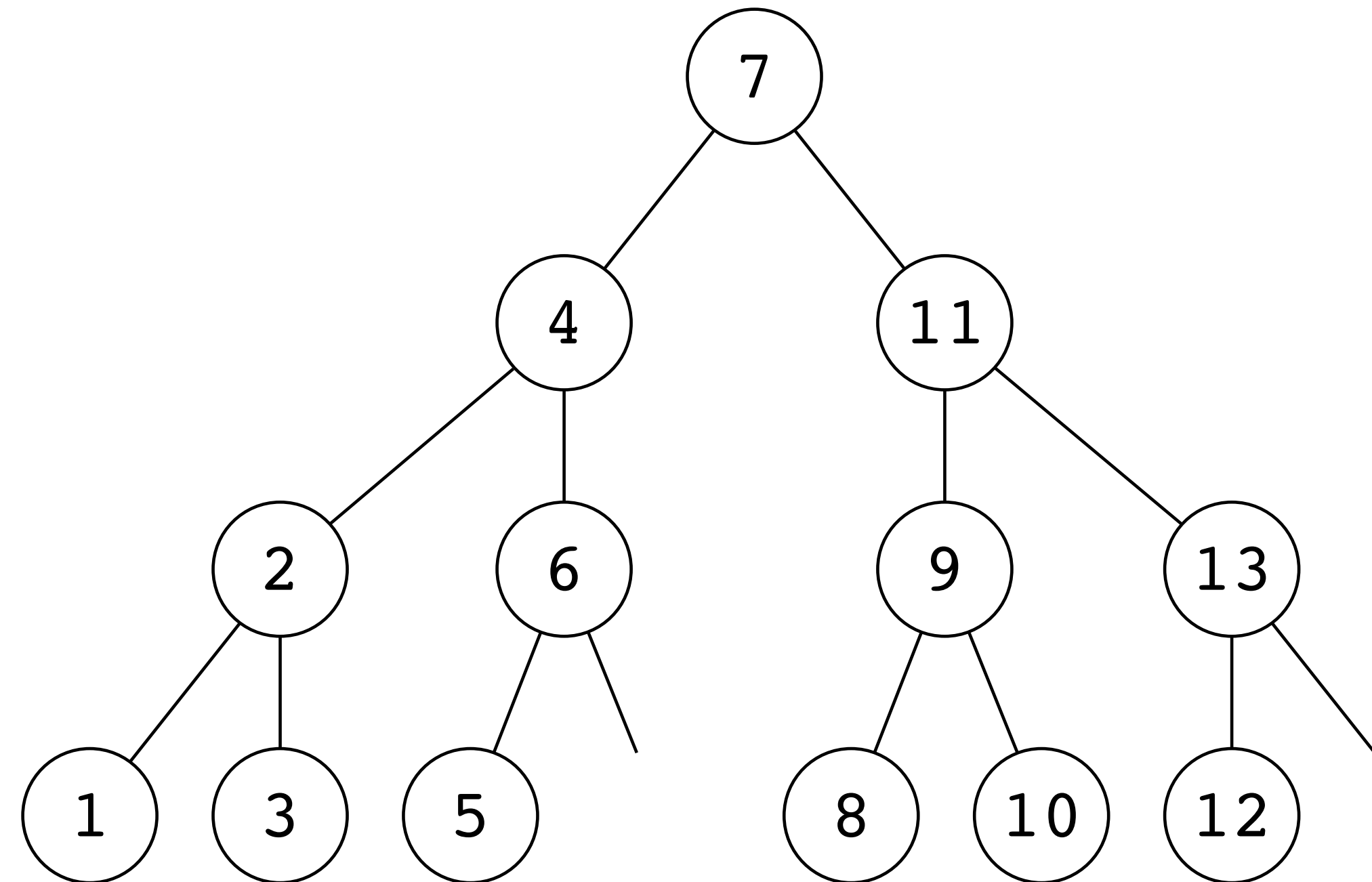
n, h et p

Dégénéré
(Pire cas)



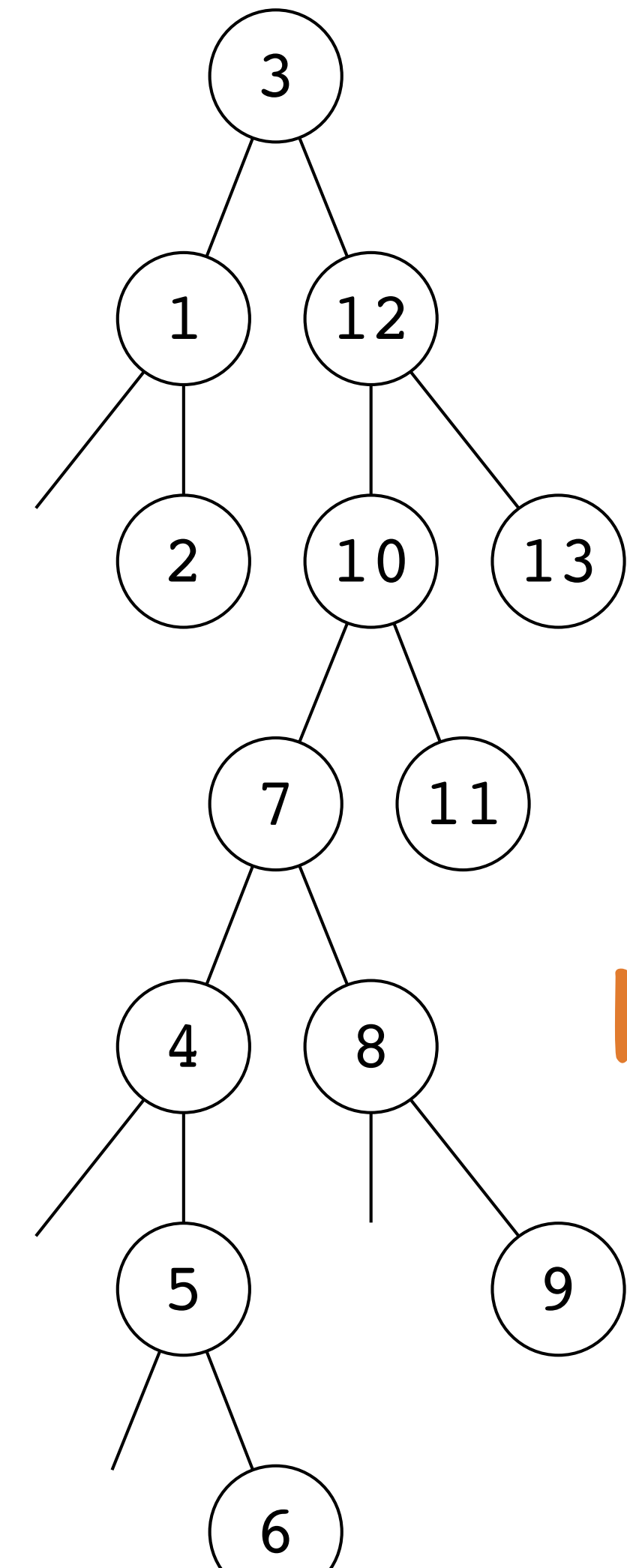
$h = O(n)$

Plein
(Meilleur cas)



$h = O(\log(n))$

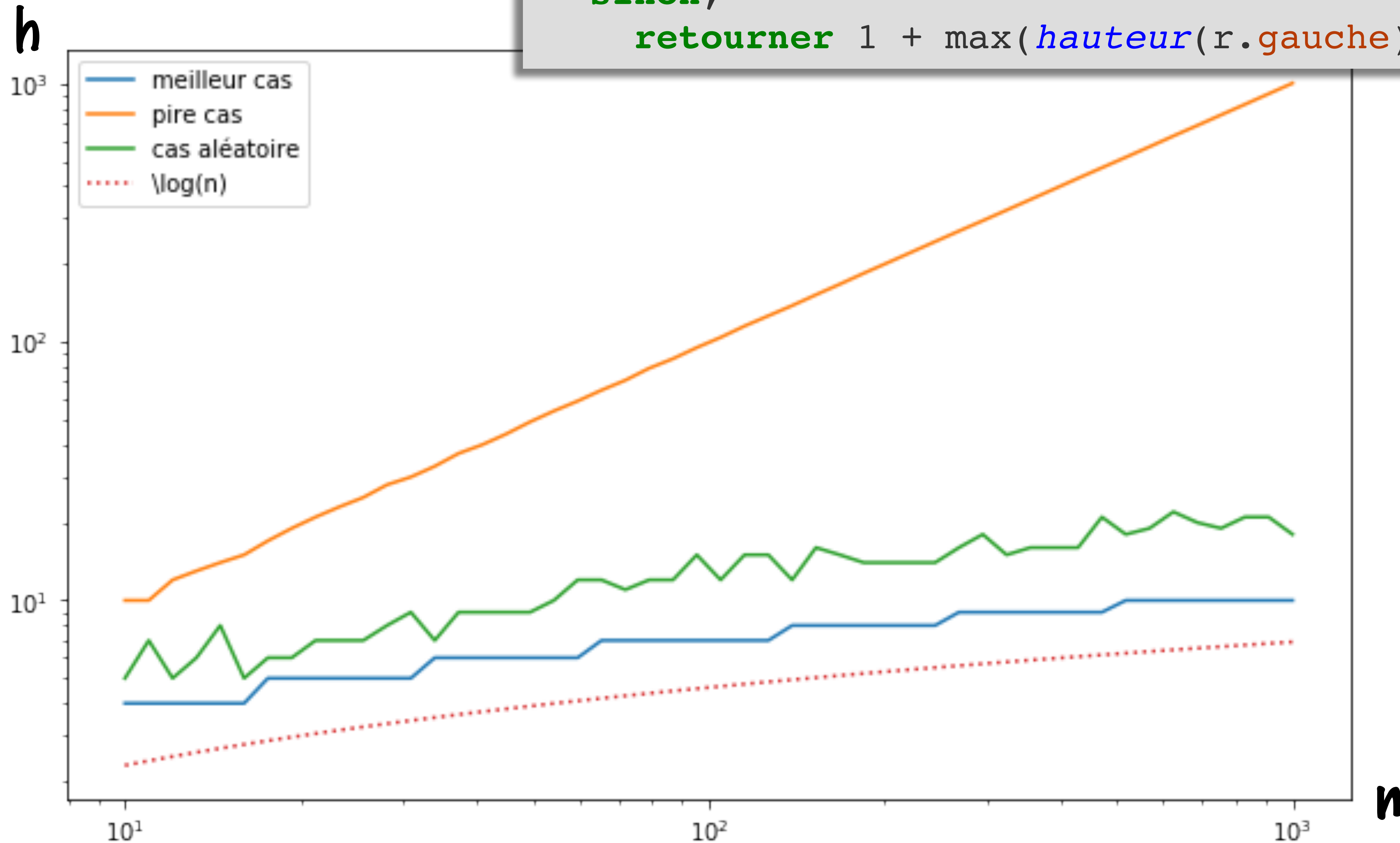
Quelconque
(Aléatoire)



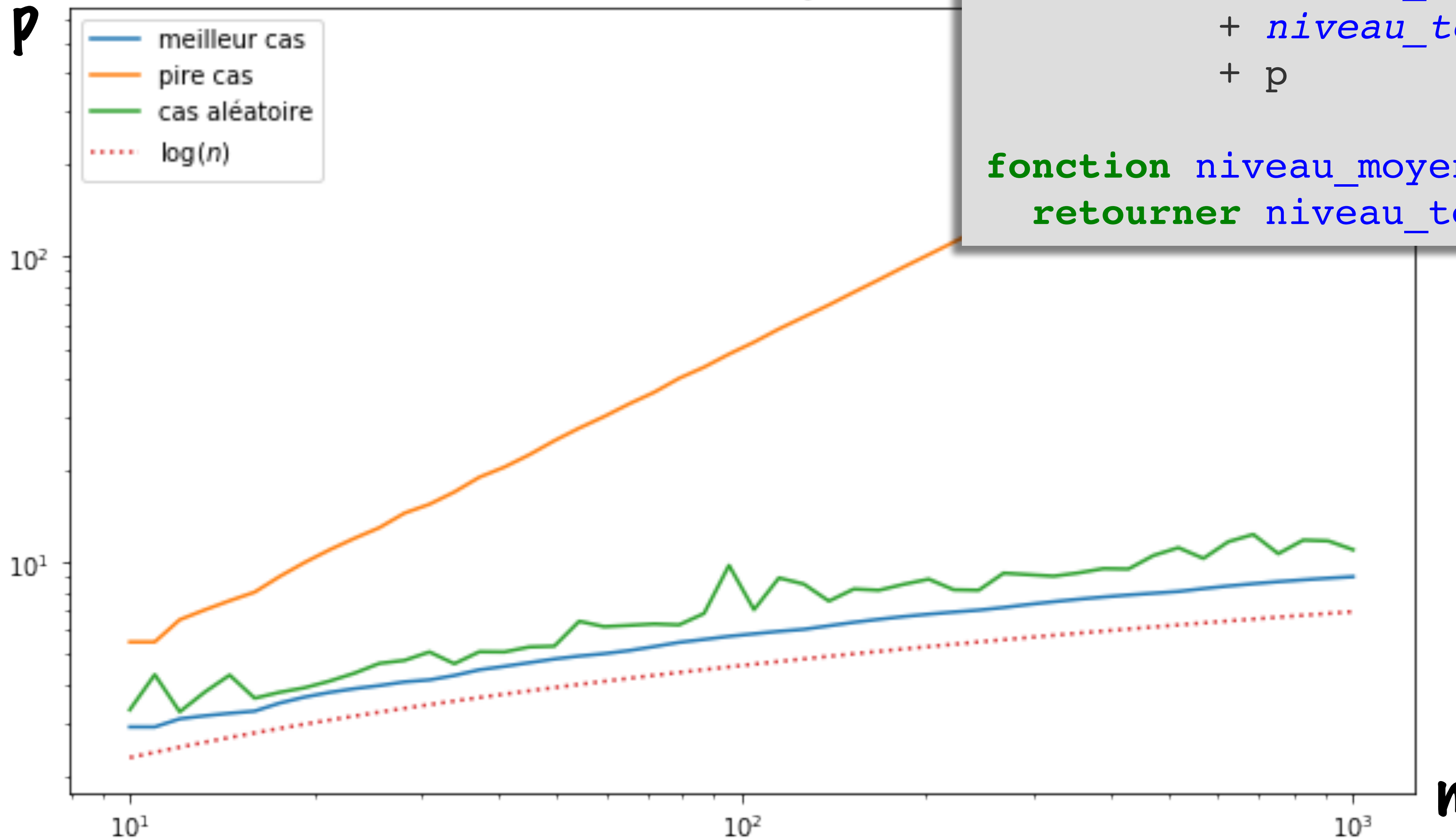
$h = O(?)$

HEIG^{VD}

```
fonction hauteur (r)
    si r == ∅,
        retourner 0
    sinon,
        retourner 1 + max(hauteur(r.gauche), hauteur(r.droit))
```



Niveaux



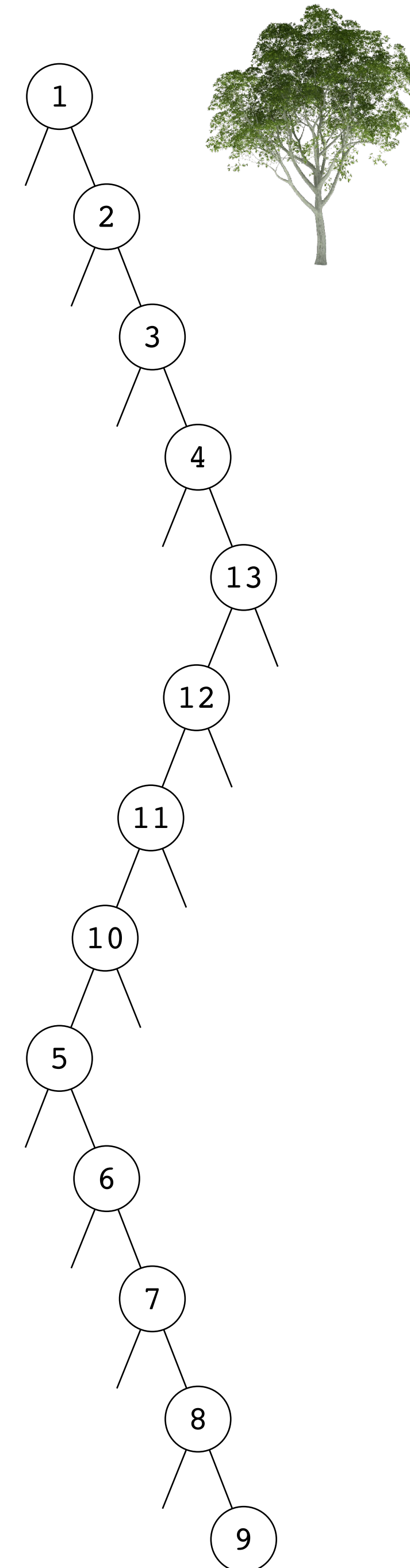
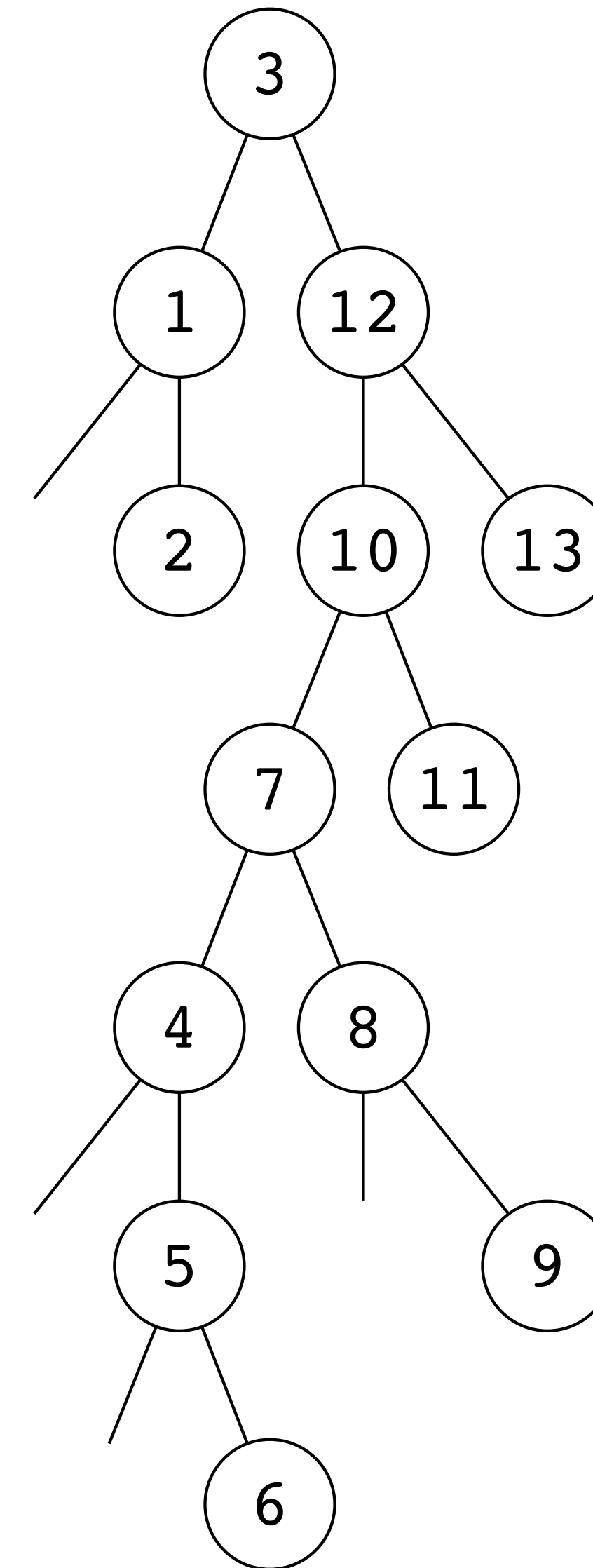
```

fonction niveau_total (r, p)
    si r == ∅,
        retourner 0
    sinon,
        retourner niveau_total(r.gauche, p+1)
                    + niveau_total(r.droit, p+1)
                    + p

fonction niveau_moyen (r)
    retourner niveau_total(r, 1) / taille(r)
    
```

$O(\log(n))$

- En moyenne, h et p sont d'ordre $\log(n)$
- Au pire, ils sont de l'ordre de n
- ...
- Mais le pire cas est un cas courant
 - Entrée triée
 - Entrée à l'envers
 - Entrées triées par morceaux
 - ...



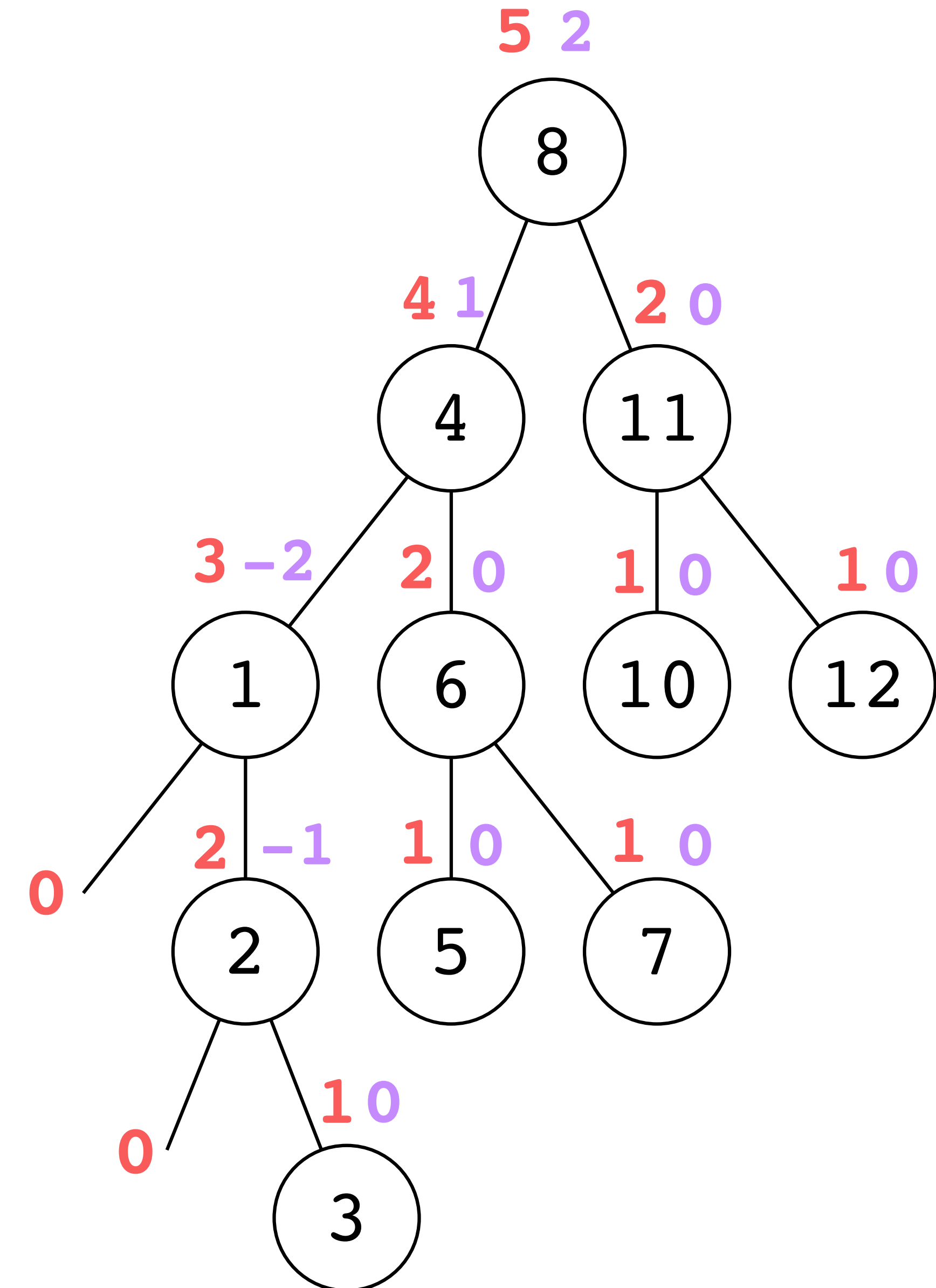


- Équilibre d'un noeud

```
fonction équilibre (r)
  si r == ∅, retourner 0
  sinon,
    retourner hauteur(r.gauche)
      - hauteur(r.droit)
```

- Un arbre est équilibré si pour tout noeud n :

$$-1 \leq \text{équilibre}(n) \leq 1$$



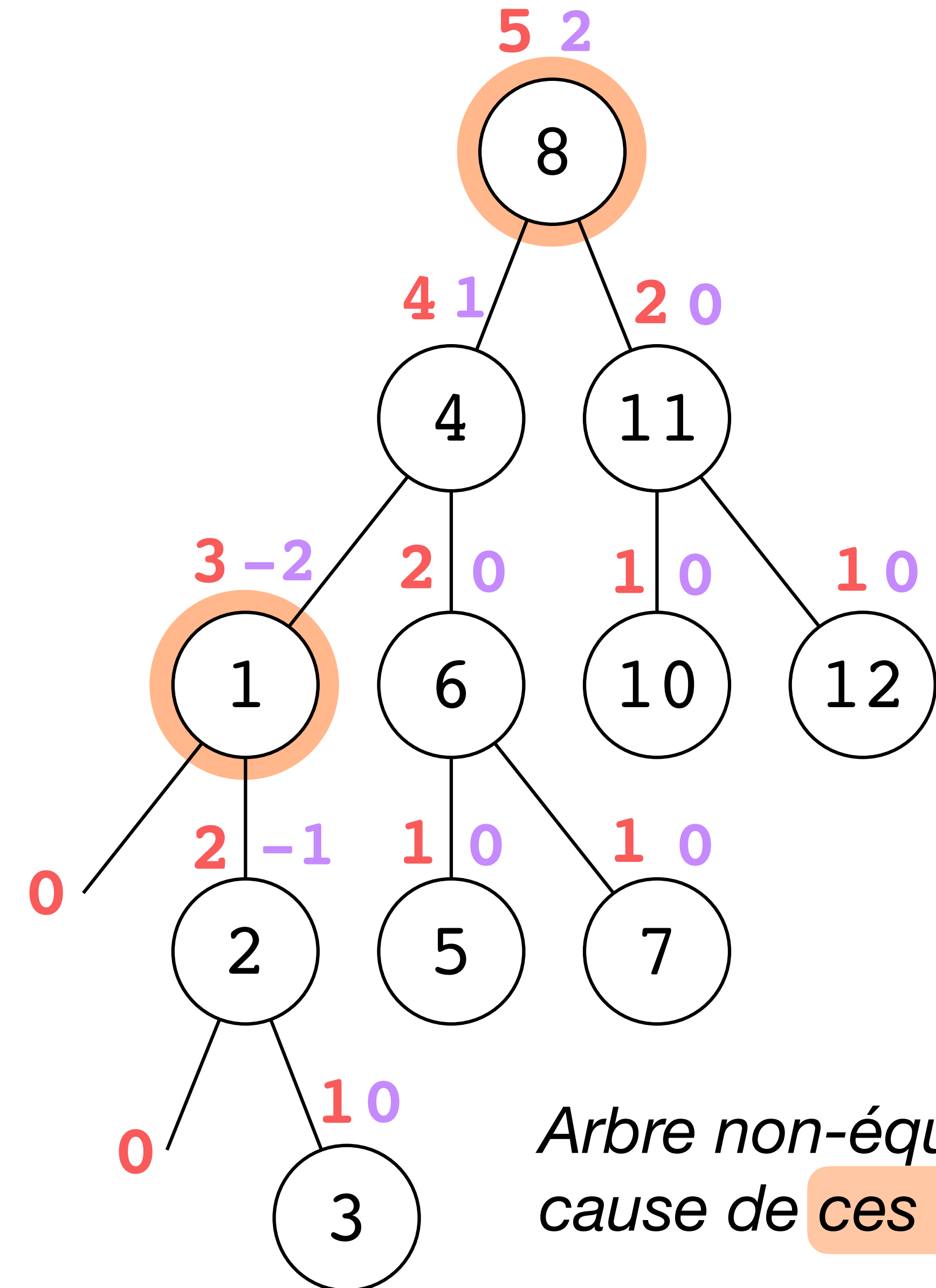


- Équilibre d'un noeud

```
fonction équilibre (r)
  si r == ∅, retourner 0
  sinon,
    retourner hauteur(r.gauche)
      - hauteur(r.droit)
```

- Un arbre est équilibré si pour tout noeud n :

$$-1 \leq \text{équilibre}(n) \leq 1$$



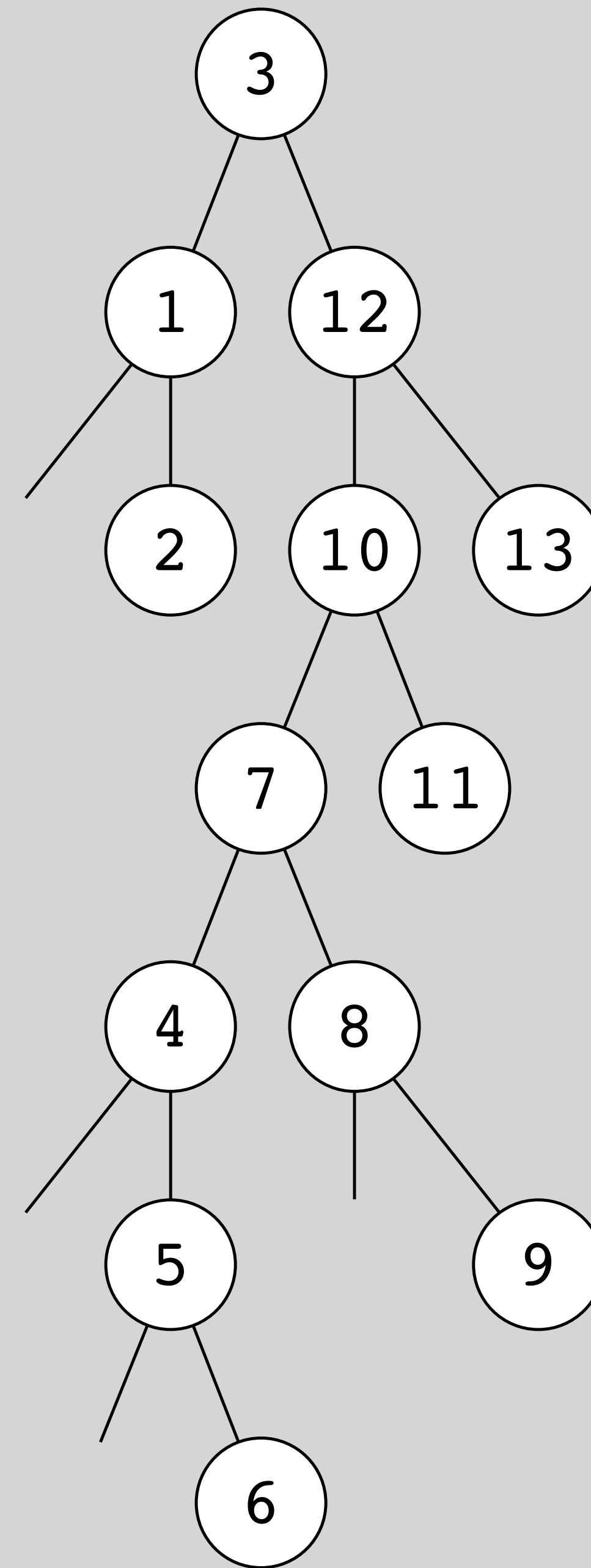


- Sur demande
- Automatiquement à chaque modification - arbres AVL, arbres rouge-noir, arbres d'Andersson, ...
- Équilibré par construction mais nombre d'enfants variables - arbres 2-3, arbres 2-3-4, b-trees, ...

Exercice



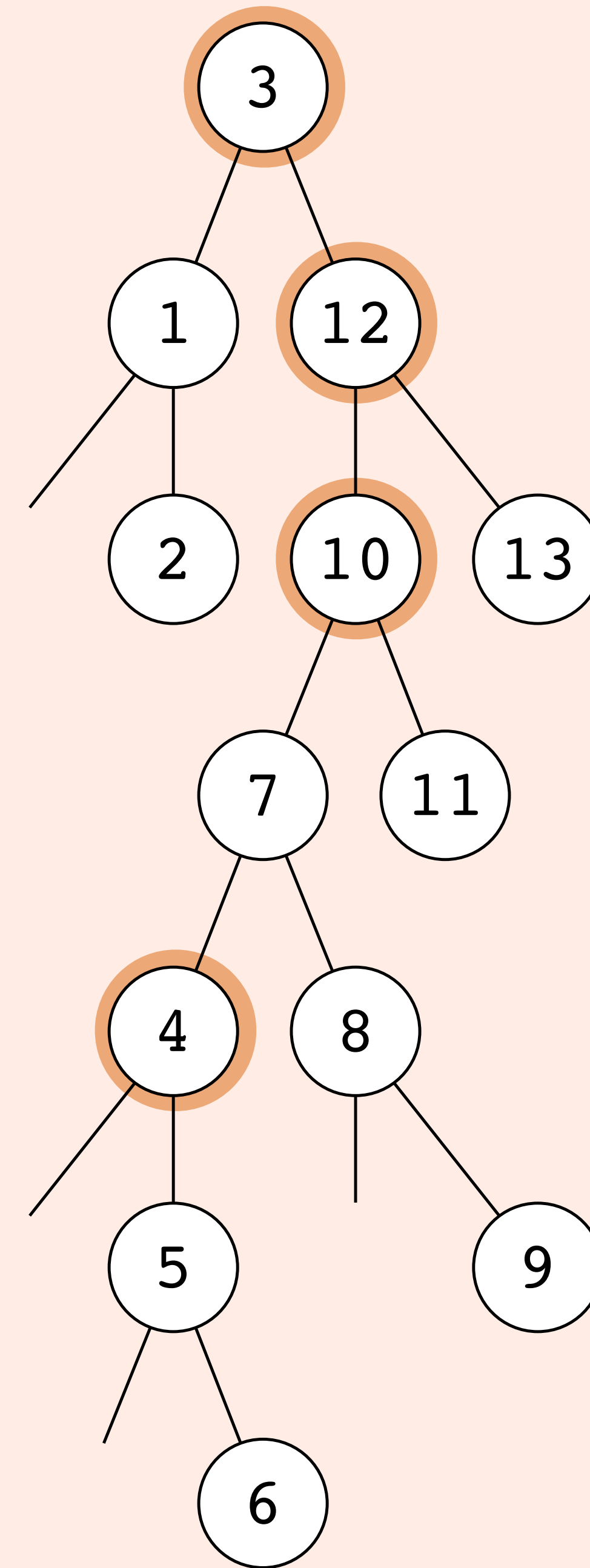
- Cet arbre est-il équilibré ?
- Si non, quels sont les noeuds où l'équilibre est rompu ?



Solution



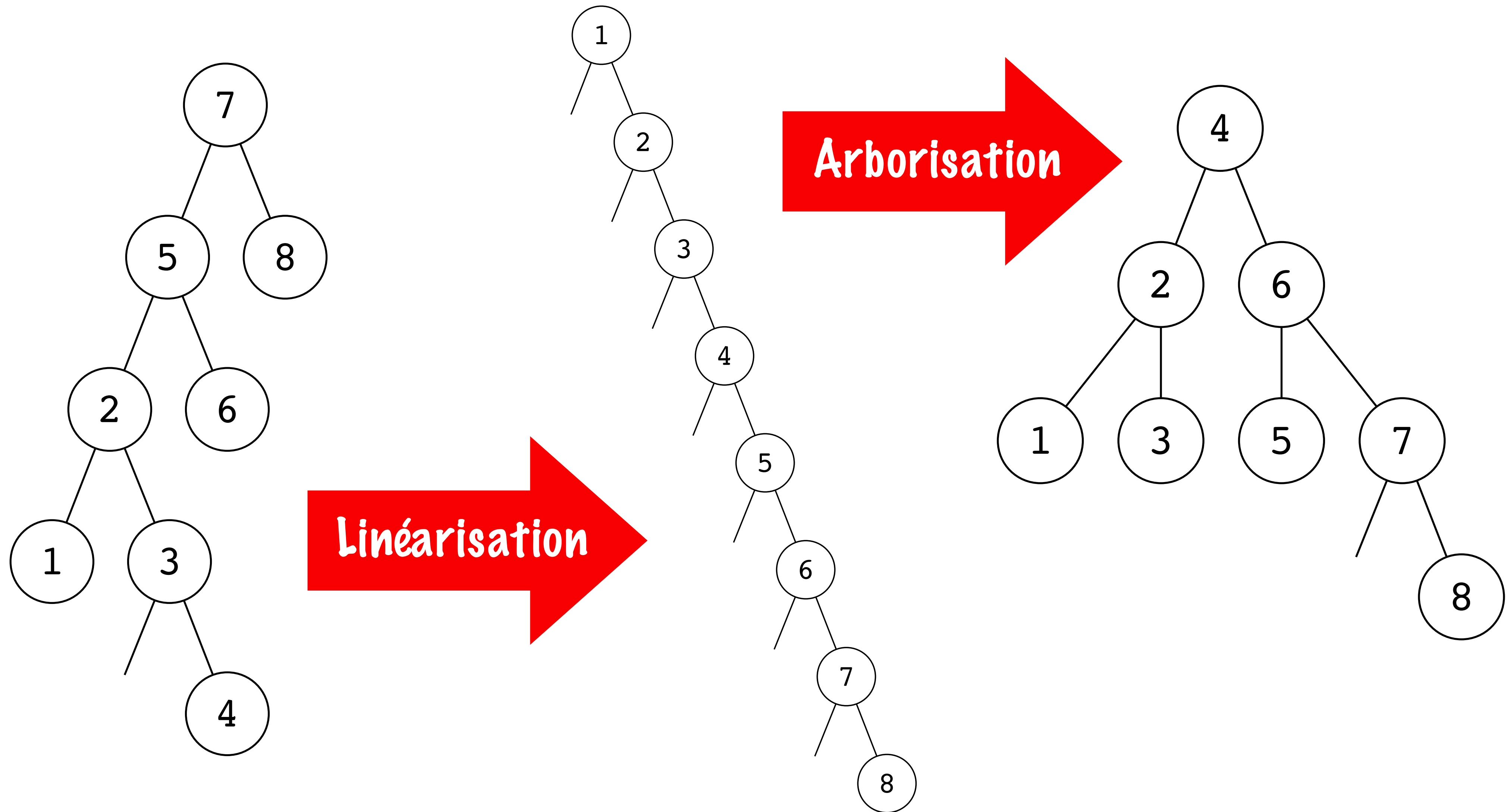
- Cet arbre n'est pas équilibré
- Les noeuds suivants rompent l'équilibre :
 - $\text{Eq}(3) = -4$
 - $\text{Eq}(12) = 4$
 - $\text{Eq}(10) = 3$
 - $\text{Eq}(4) = -2$



14 . Equilibrage par linéarisation et arborisation



Equilibrage à la demande

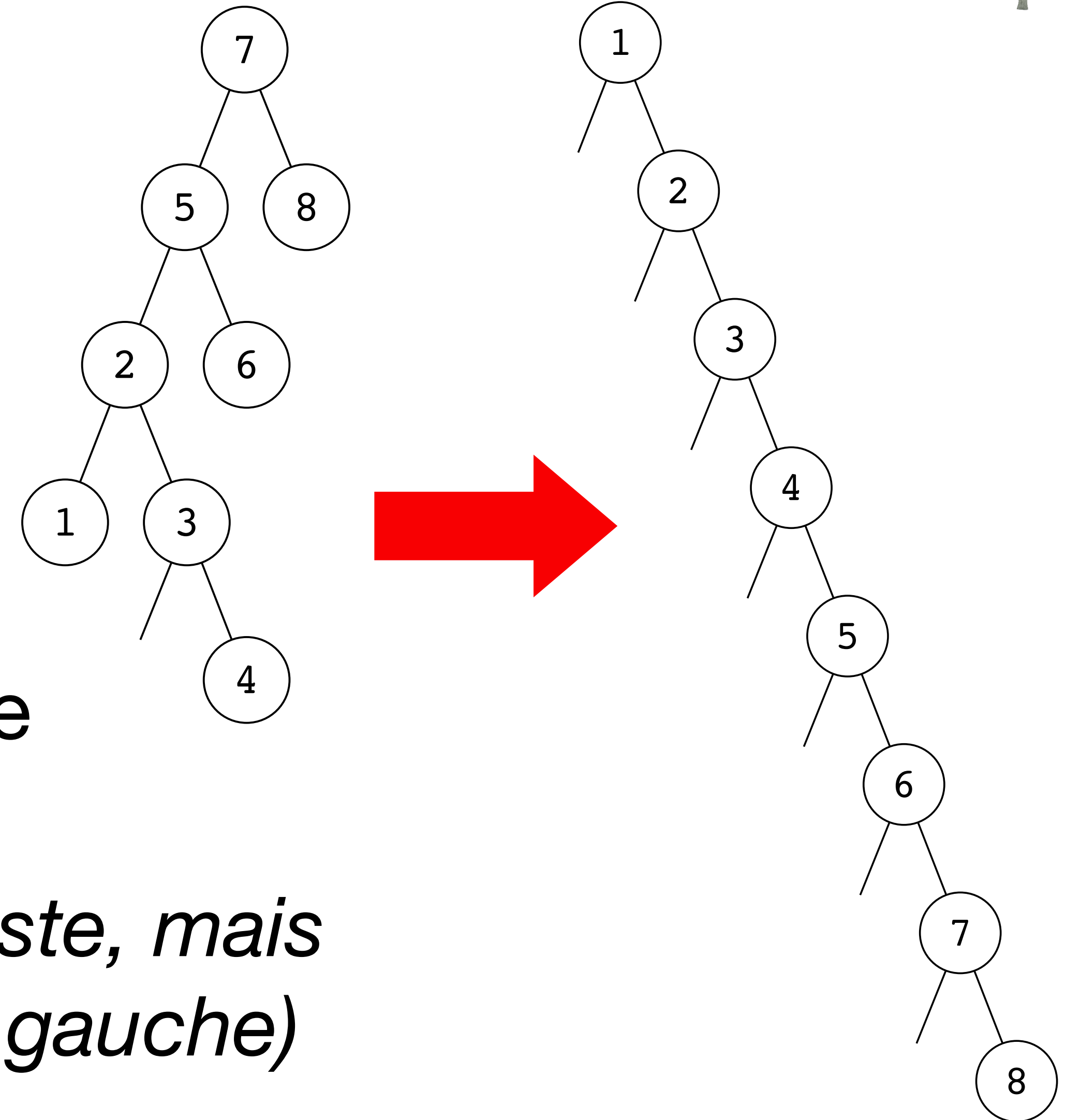


Linéarisation



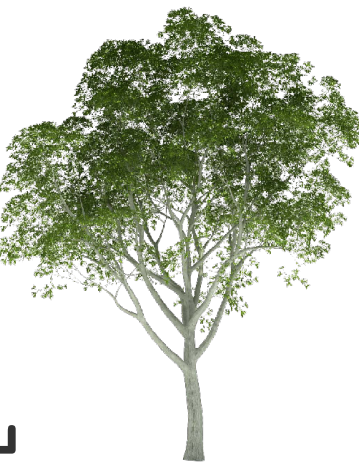
Ré-organisation de l'arbre

- Ne touche qu'aux liens
- Le résultat n'a aucun enfant gauche



(Topologiquement équivalent à une liste, mais avec des noeuds d'ABR sans enfant gauche)

Linéarisation



Paramètres :

(*in*) Lien r vers la racine de l'arbre en entrée

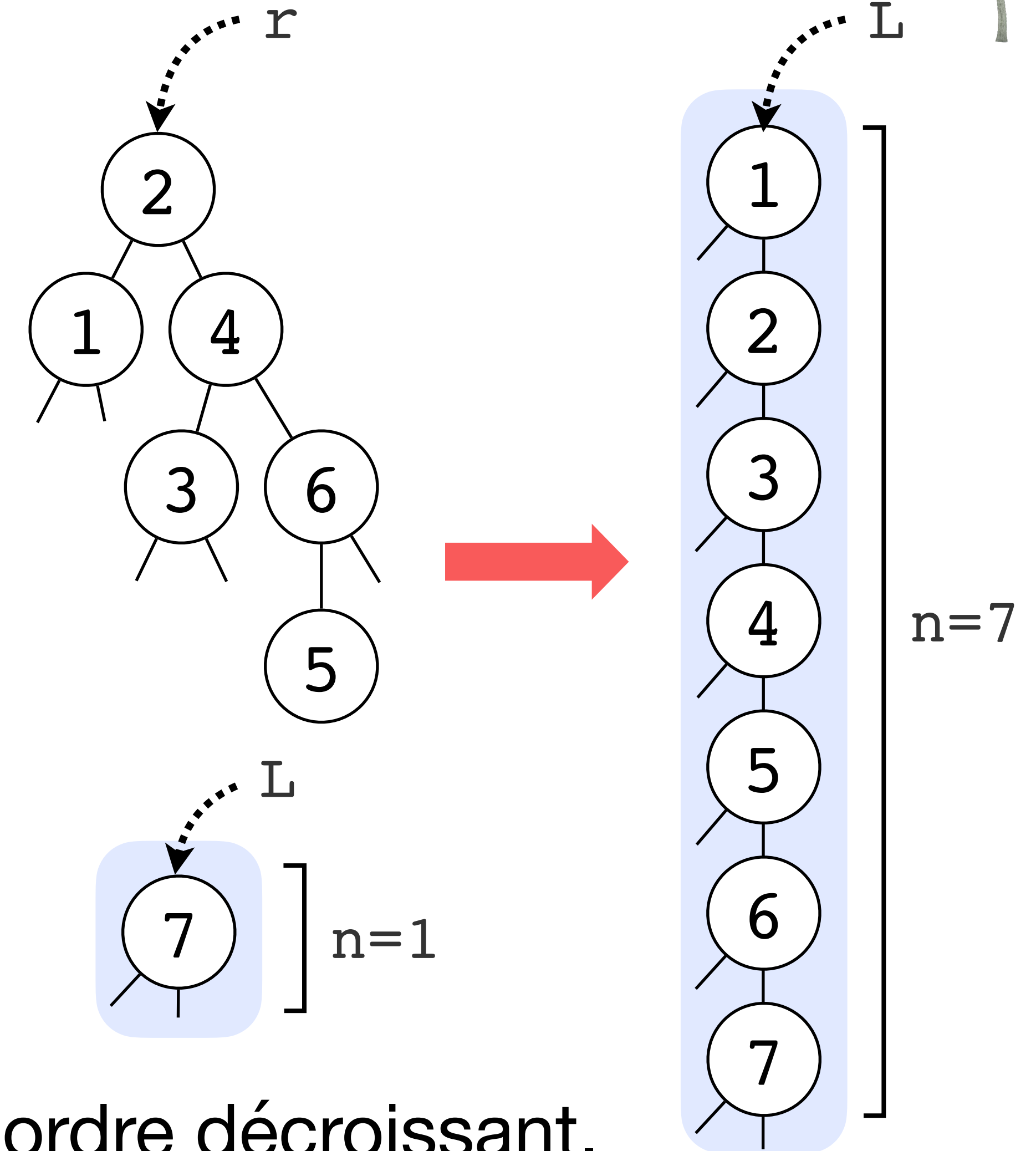
(*inout*) Lien L vers la racine de l'arbre en sortie

(*inout*) Compteur n du nombre de noeuds traités

Idée

Parcourir r pour ajouter chaque noeud à L dans l'ordre décroissant.

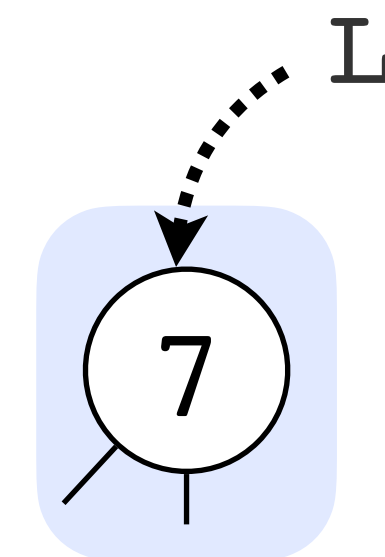
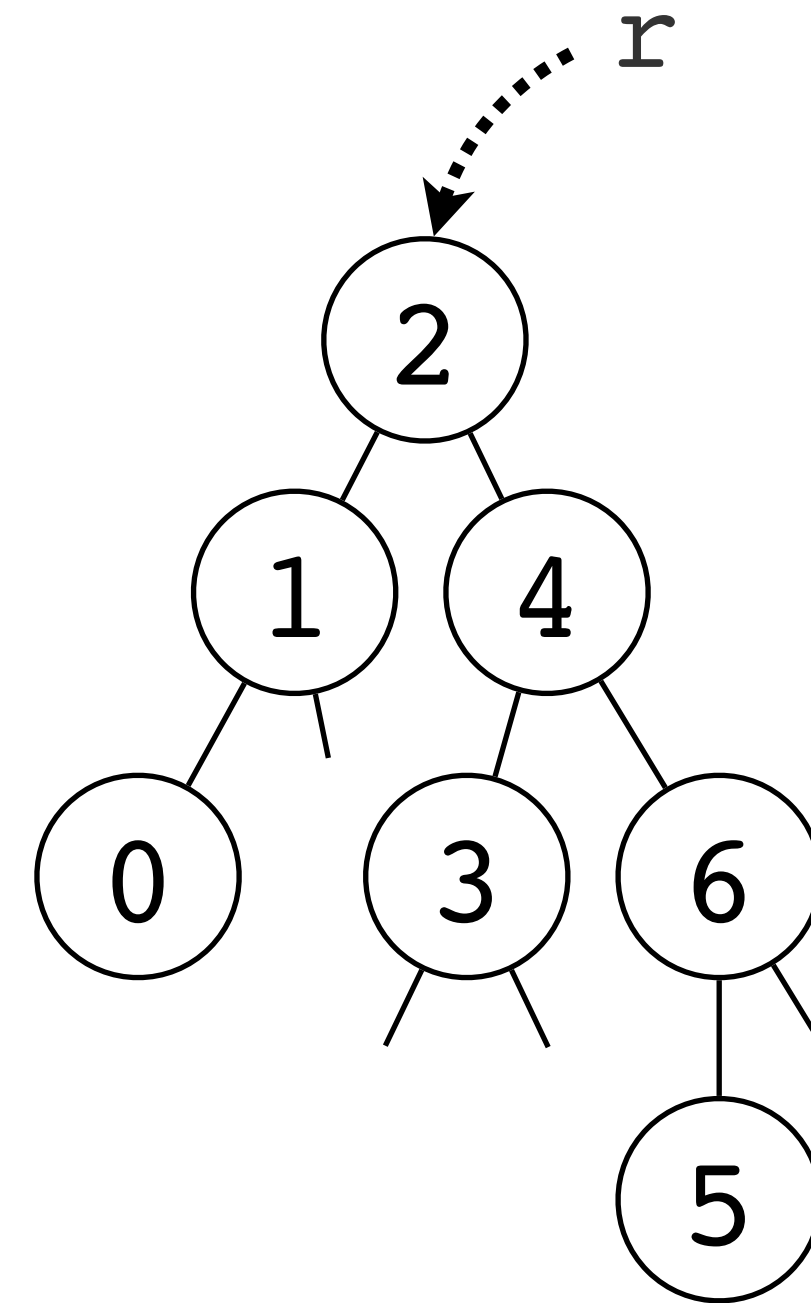
On insère en tête de L , il faut donc parcourir r en ordre symétrique inverse.



Linéarisation



```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```

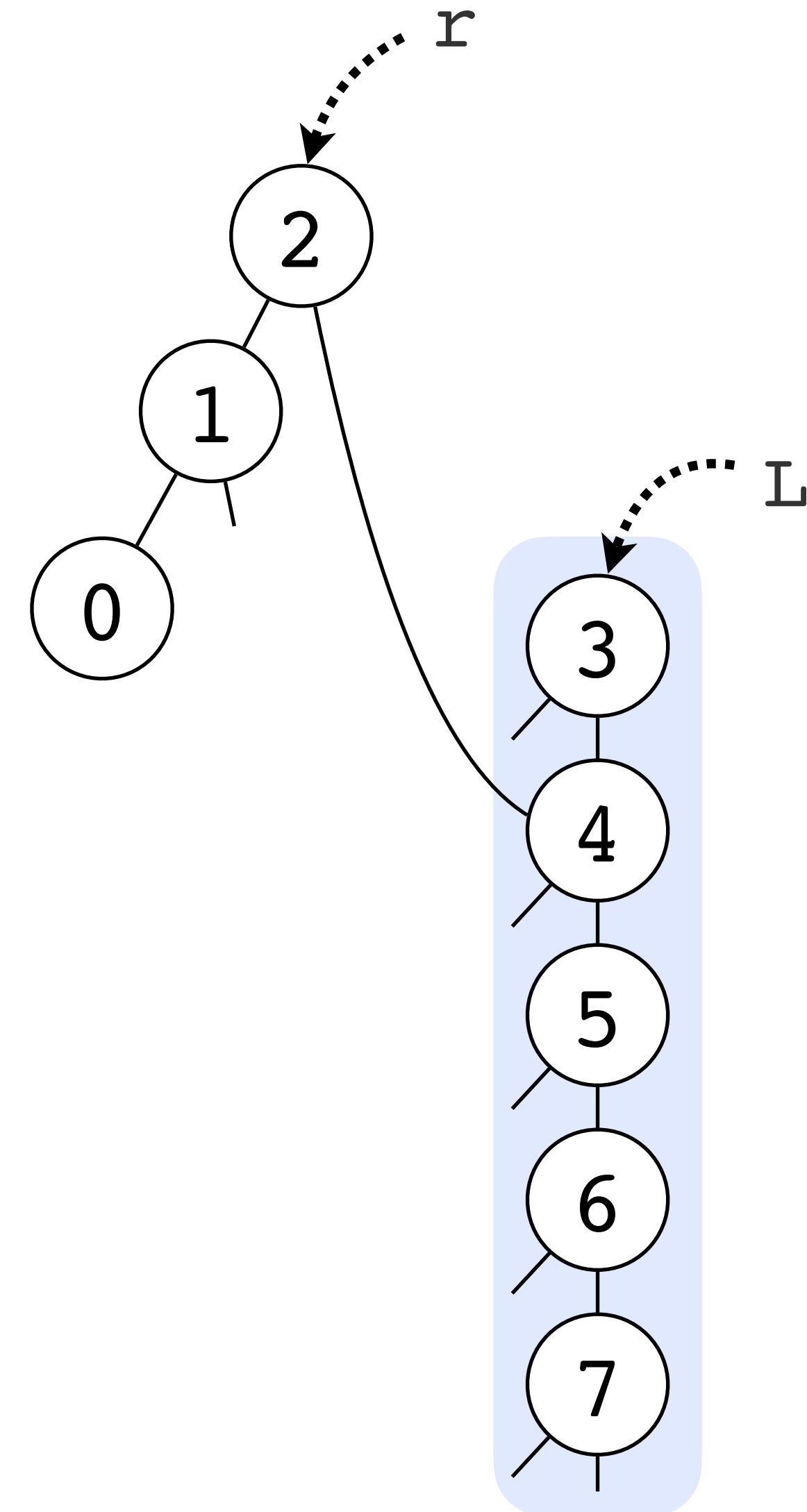


Linéarisation



```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```

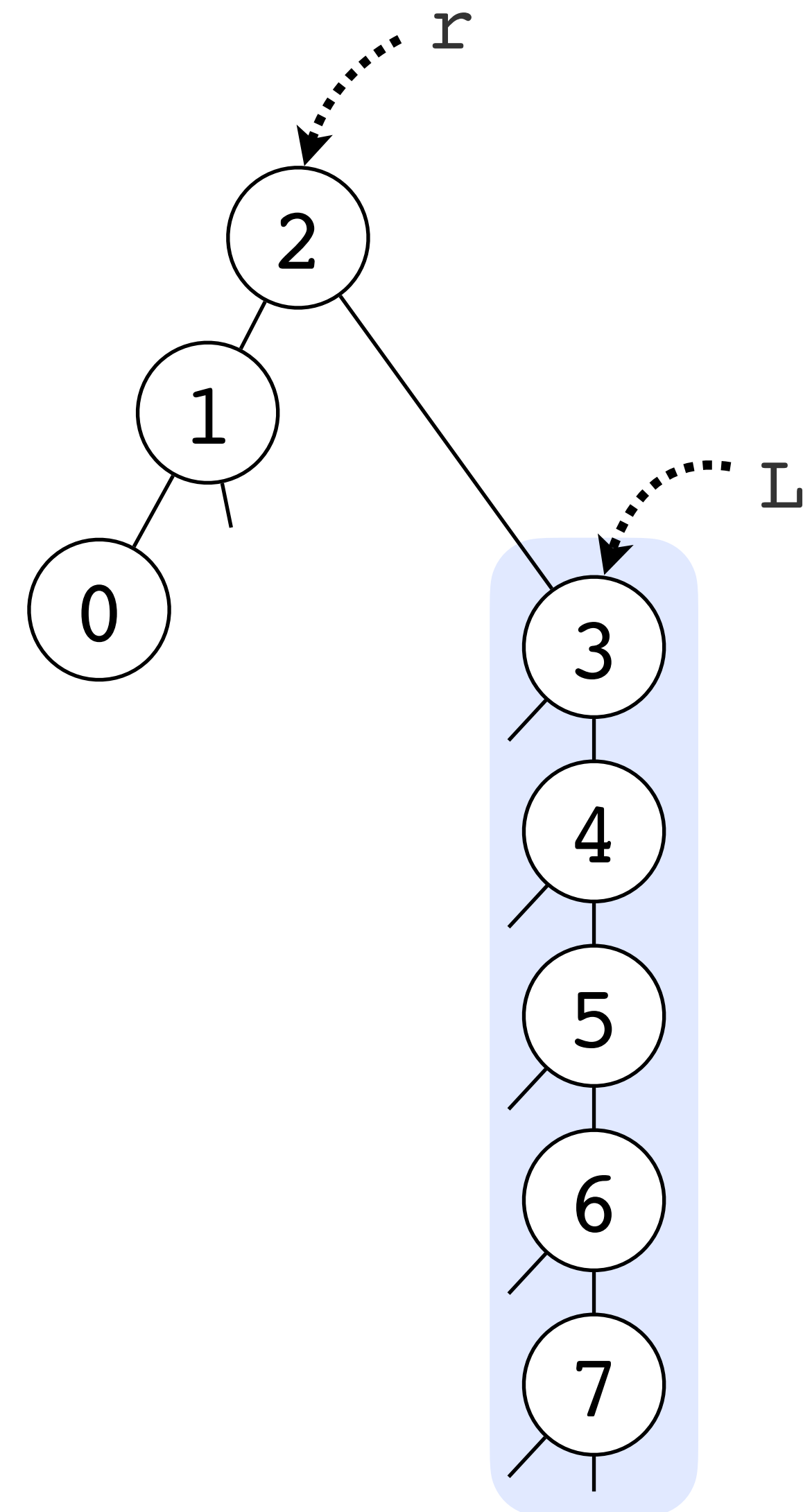
Après la première linéarisation,
L pointe sur la linéarisation de r.droit
r.droit pointe toujours sur son enfant



Linéarisation



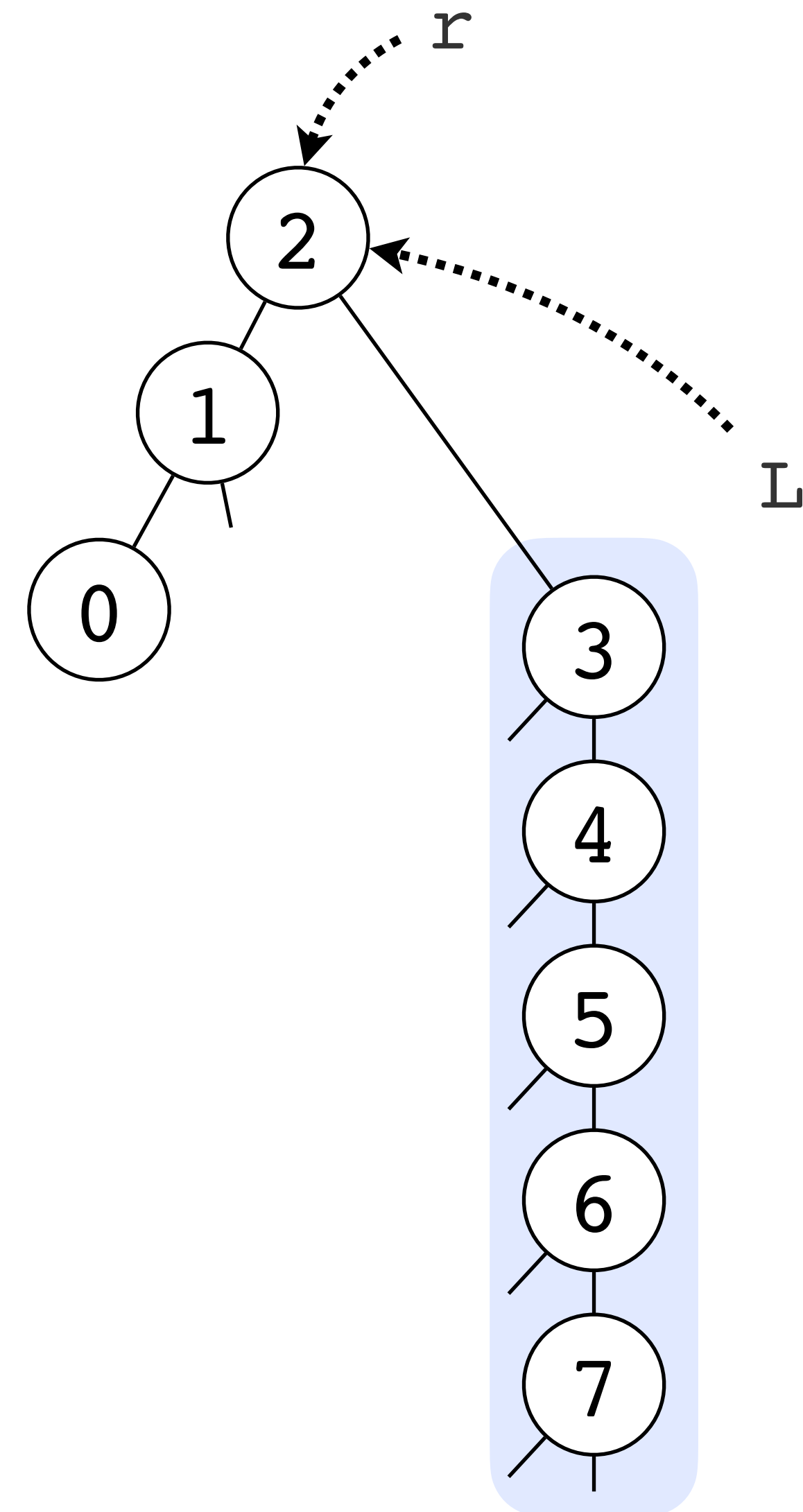
```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```



Linéarisation



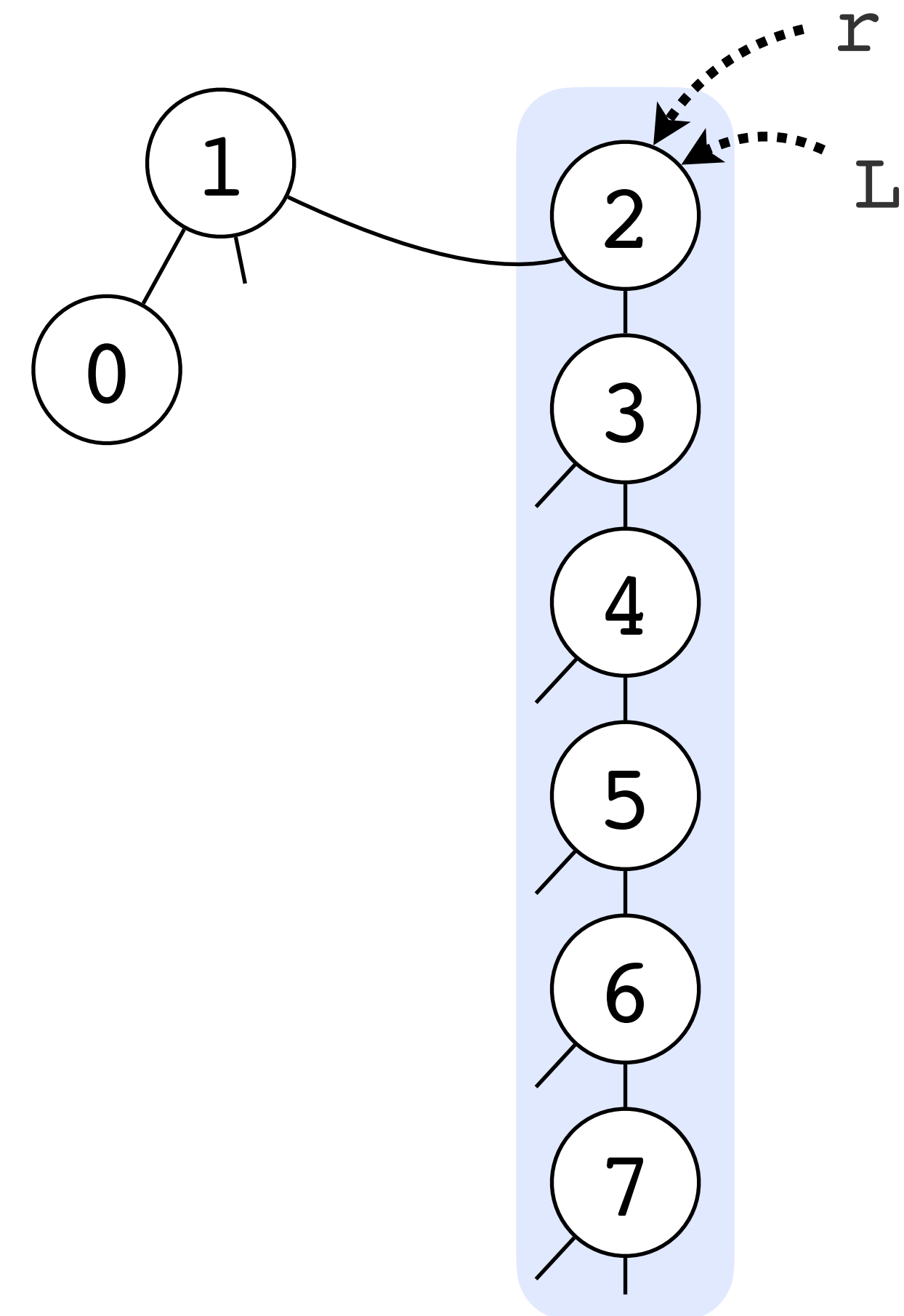
```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```





```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```

Entre les deux linéarisations,
r est ajouté à L



*(Déplacement de la racine pour clarifier la visualisation.
Pas de changement structurel)*

Linéarisation



```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```

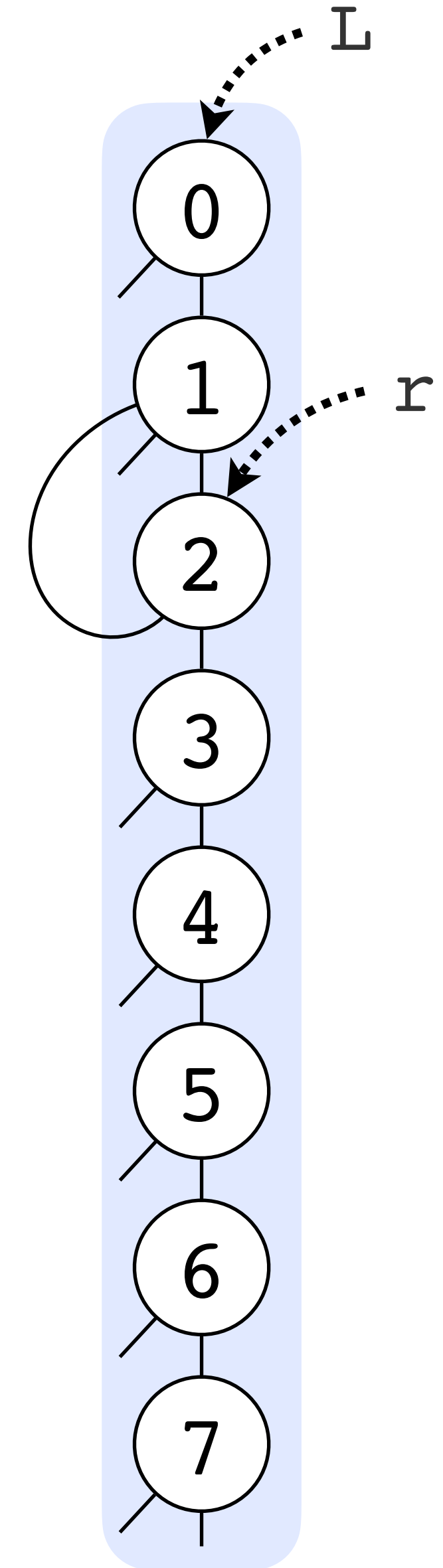
Avant la seconde linéarisation

L pointe sur la linéarisation de `r.droit` et `r`.

Après la seconde linéarisation,

L pointe sur la linéarisation de tout l'arbre

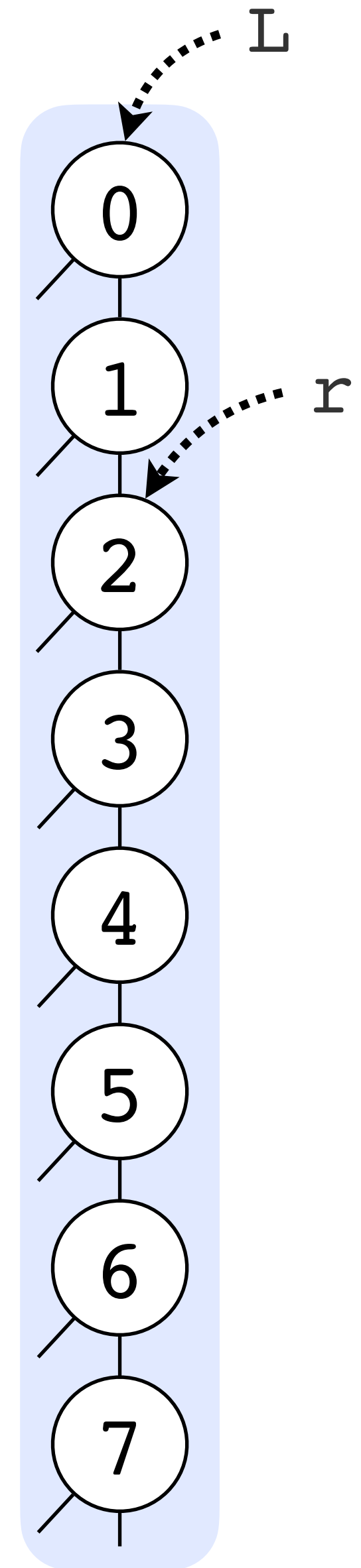
`r.gauche` pointe toujours sur son enfant



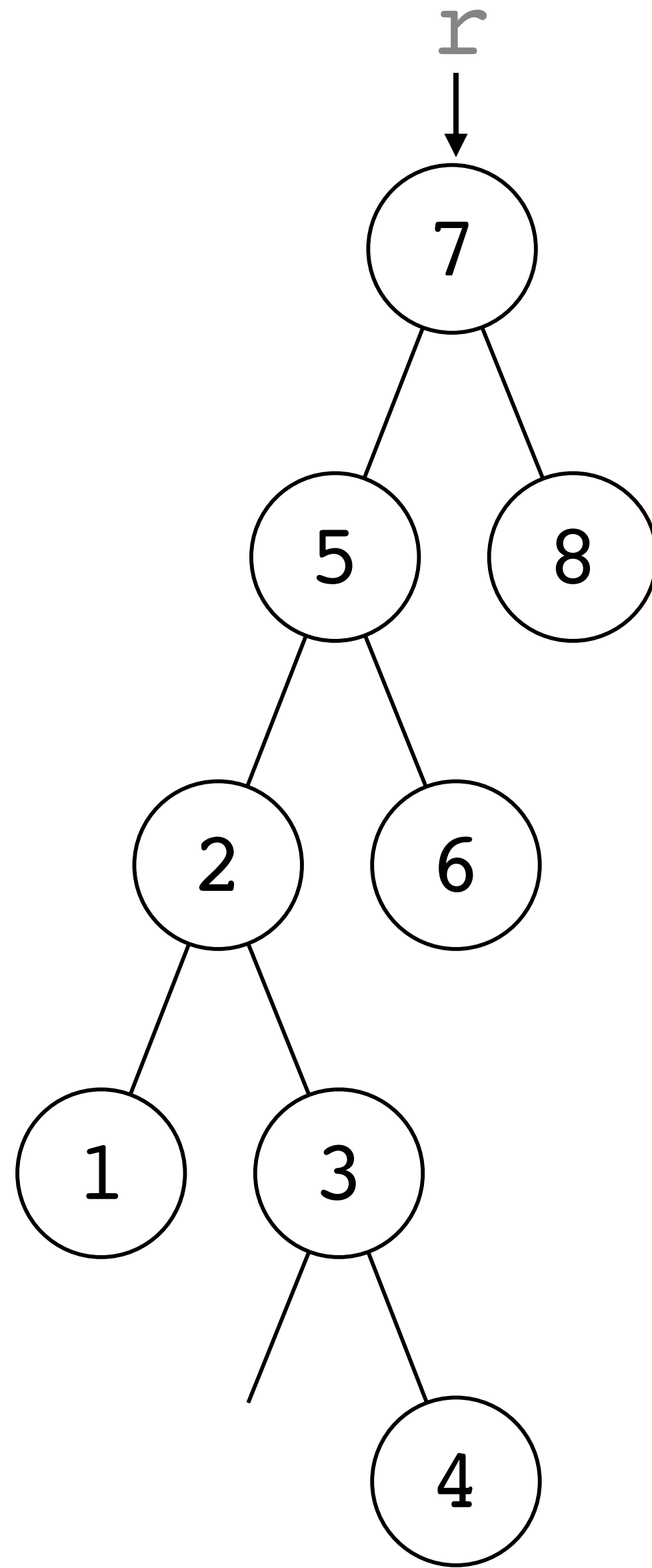
Linéarisation



```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```



Linéariser



```
linéariser(r=7, L=∅, n=0)
```

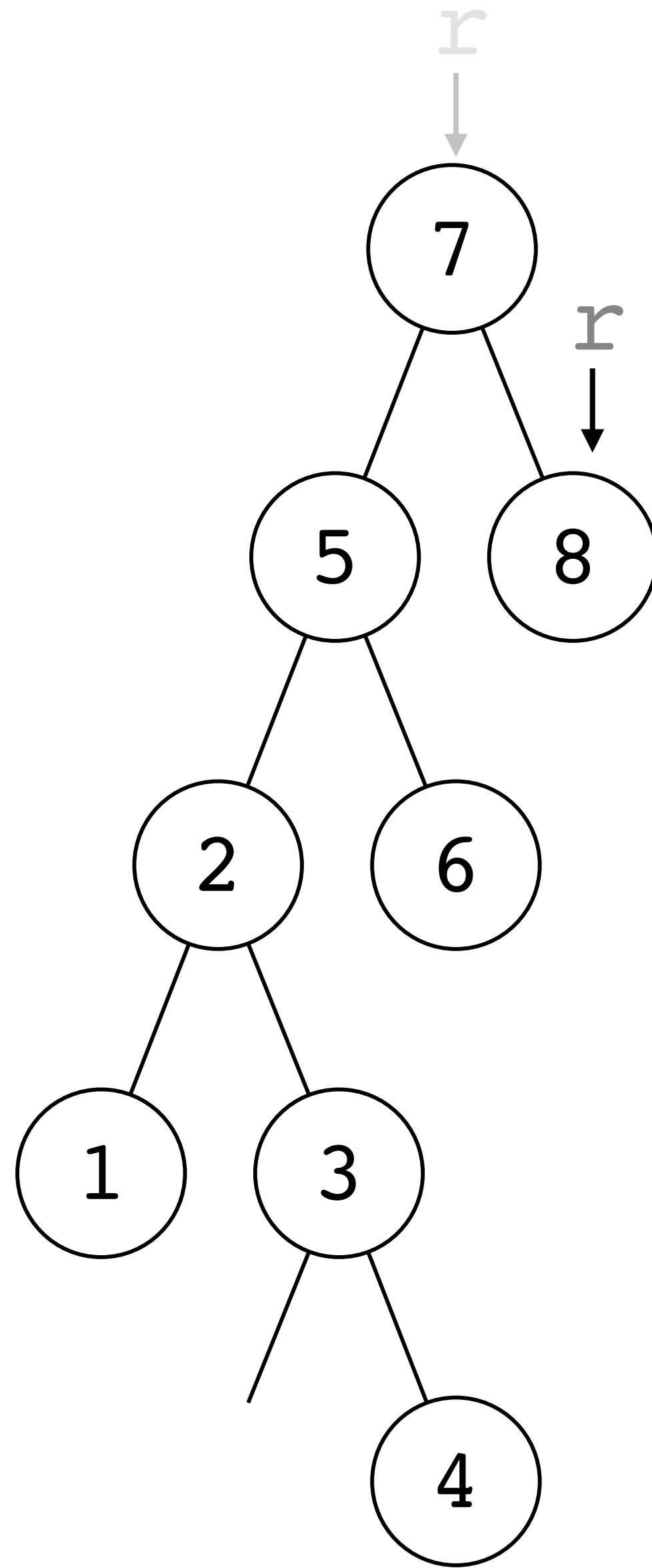
```
linéariser(r=8, L=∅, n=0)
```

```
8.droit ← ∅, L ← 8, n ← 1
```

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
  
```

Linéariser



```
linéariser(r=7, L=∅, n=0)
```

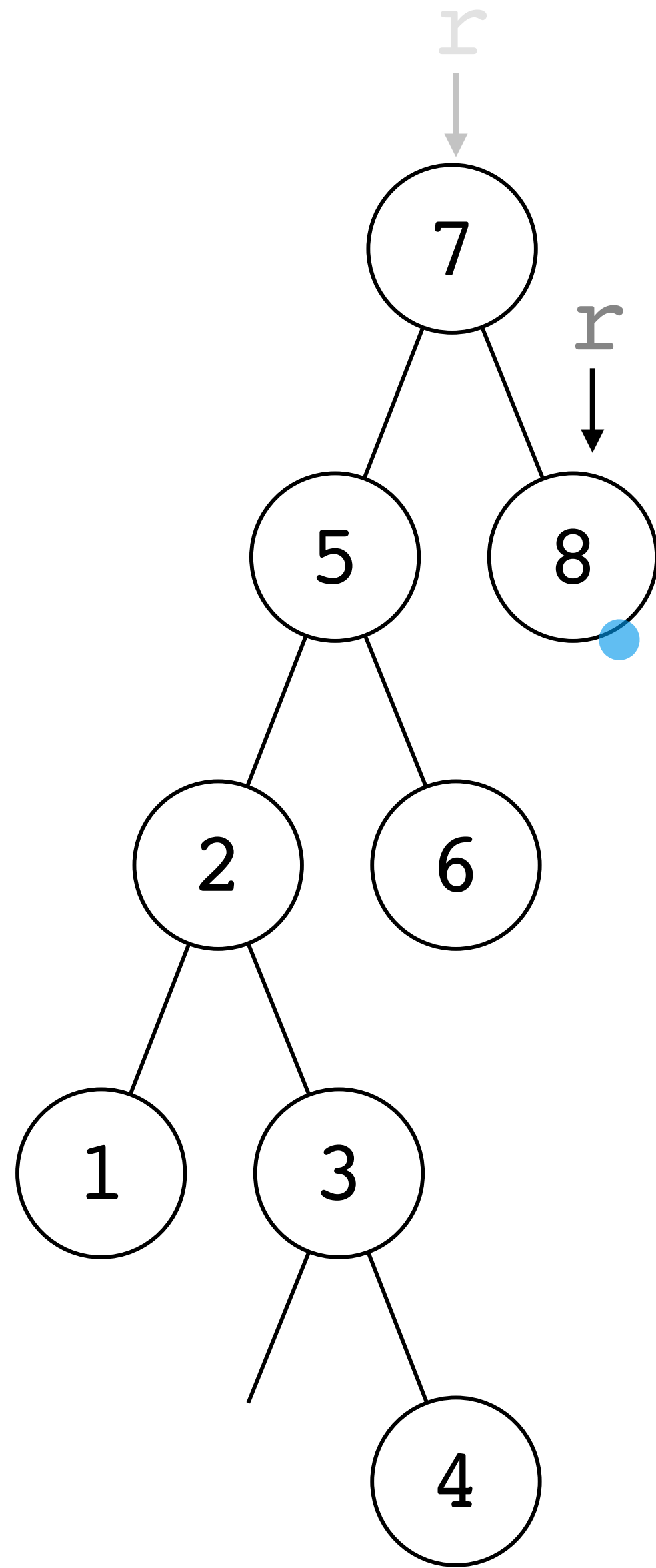
```
linéariser(r=8, L=∅, n=0)
```

```
8.droit ← ∅, L ← 8, n ← 1
```

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
    
```

Linéariser



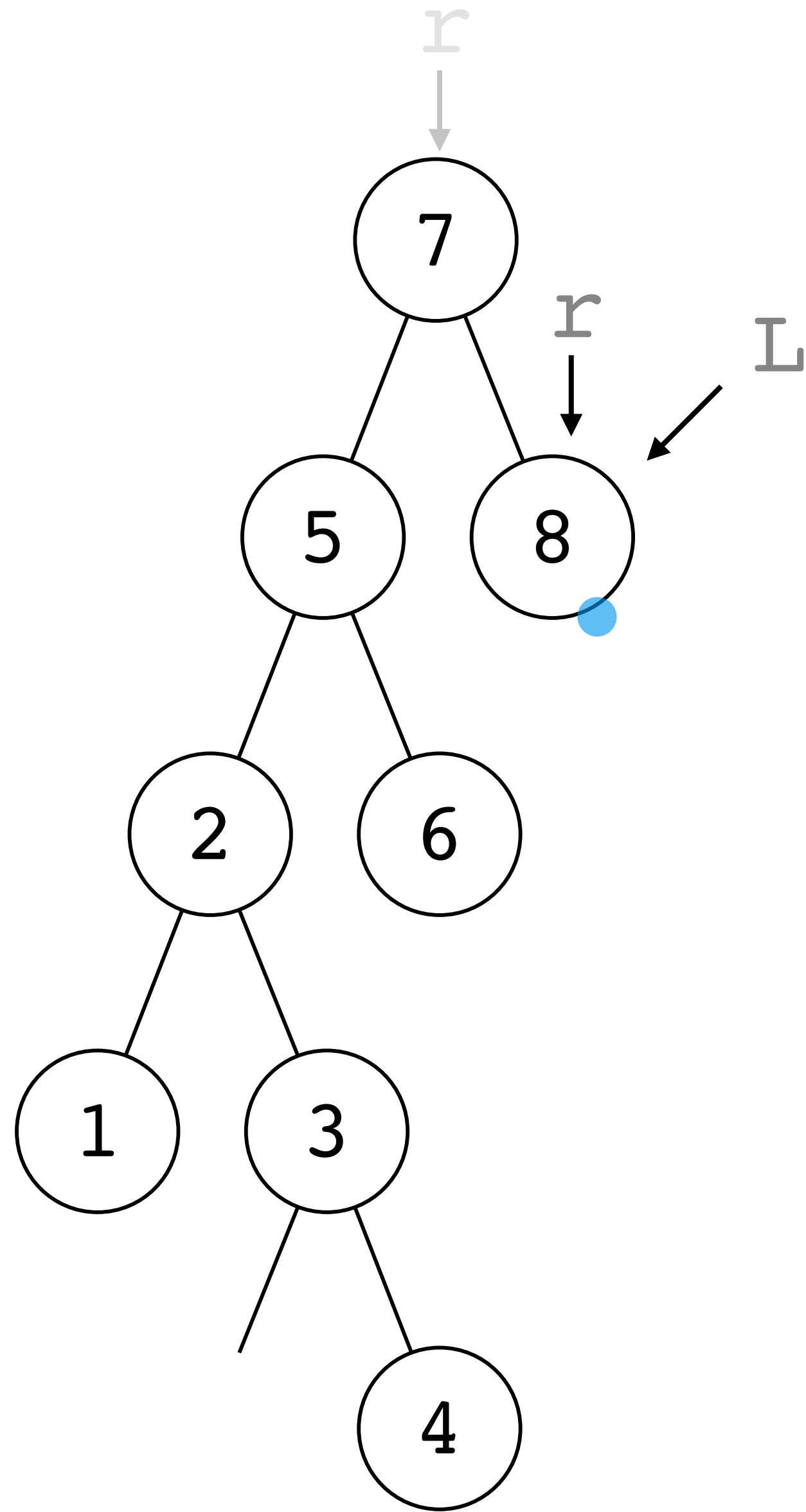
```
linéariser(r=7, L=∅, n=0)
```

```
linéariser(r=8, L=∅, n=0)
```

```
8.droit ← ∅, L ← 8, n ← 1
```

```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```

Linéariser

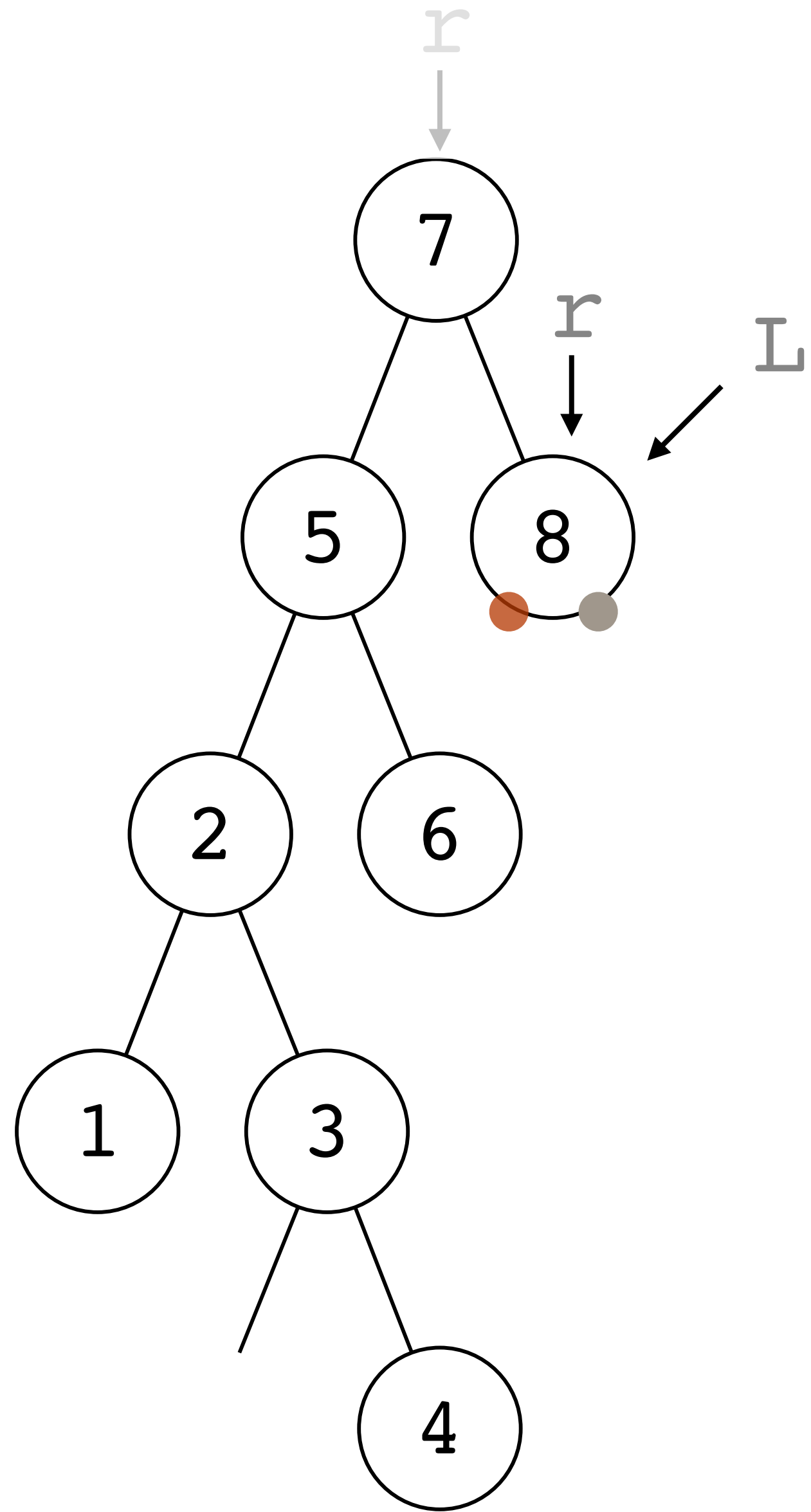


```
linéariser(r=7, L=∅, n=0)
```

```
linéariser(r=8, L=∅, n=0)
```

```
8.droit ← ∅, L ← 8, n ← 1
```

```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```

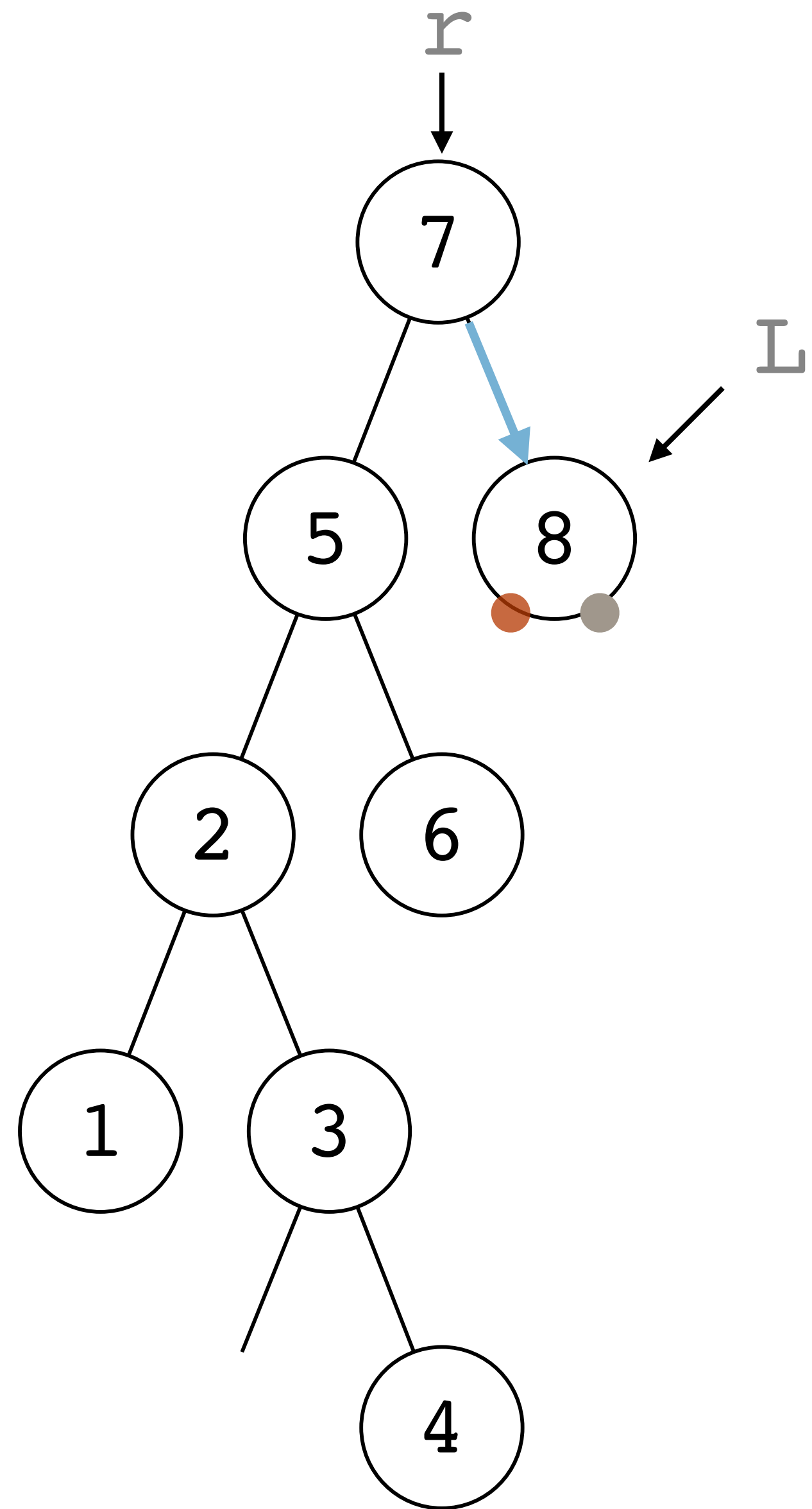


```

linéariser(r=7, L=∅, n=0)
  linéariser(r=8, L=∅, n=0)
    8.droit ← ∅, L ← 8, n ← 1
    8.gauche ← ∅
  7.droit ← 8, L ← 7, n ← 2
  
```

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
  
```

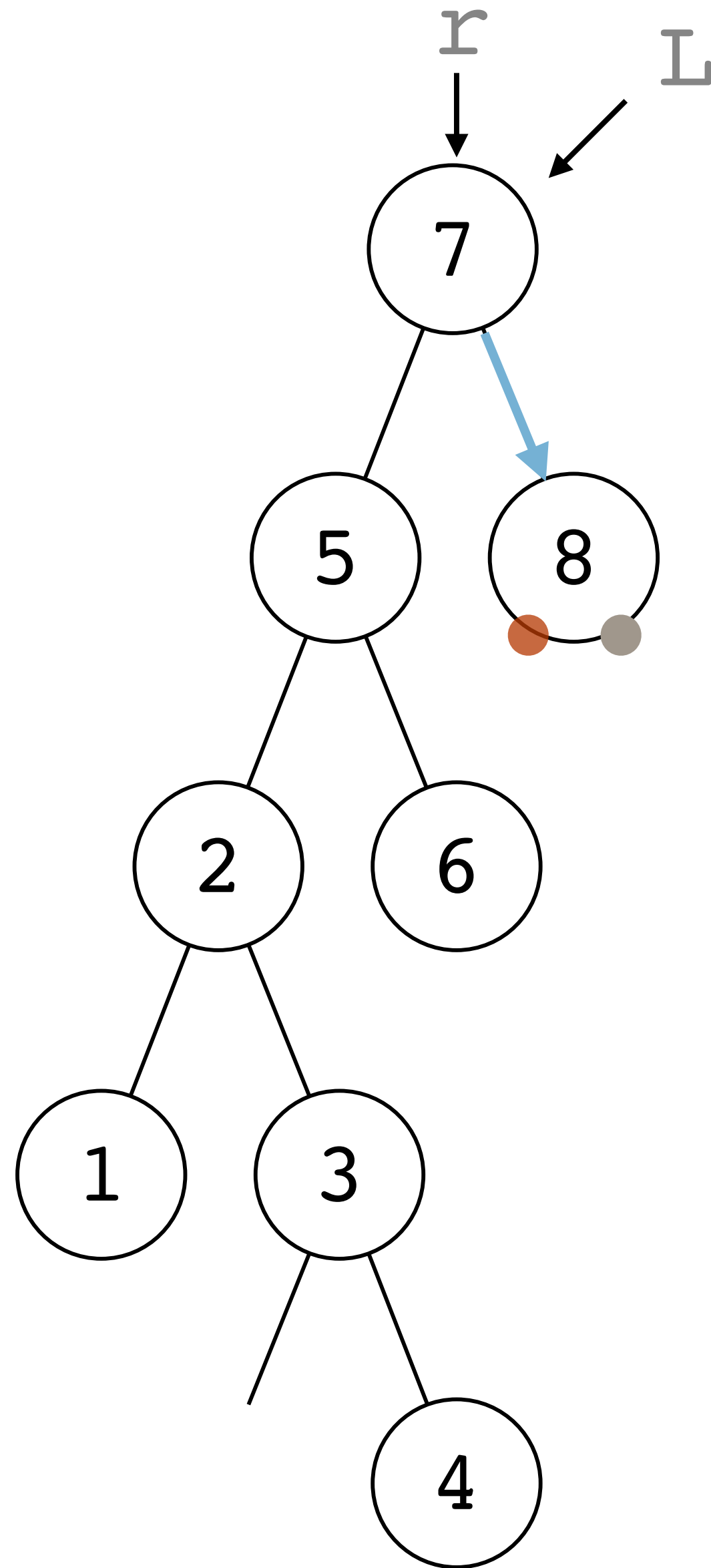


```

linéariser(r=7, L=∅, n=0)
  linéariser(r=8, L=∅, n=0)
    8.droit ← ∅, L ← 8, n ← 1
    8.gauche ← ∅
  7.droit ← 8, L ← 7, n ← 2
  
```

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
  
```



```
linéariser(r=7, L=∅, n=0)
```

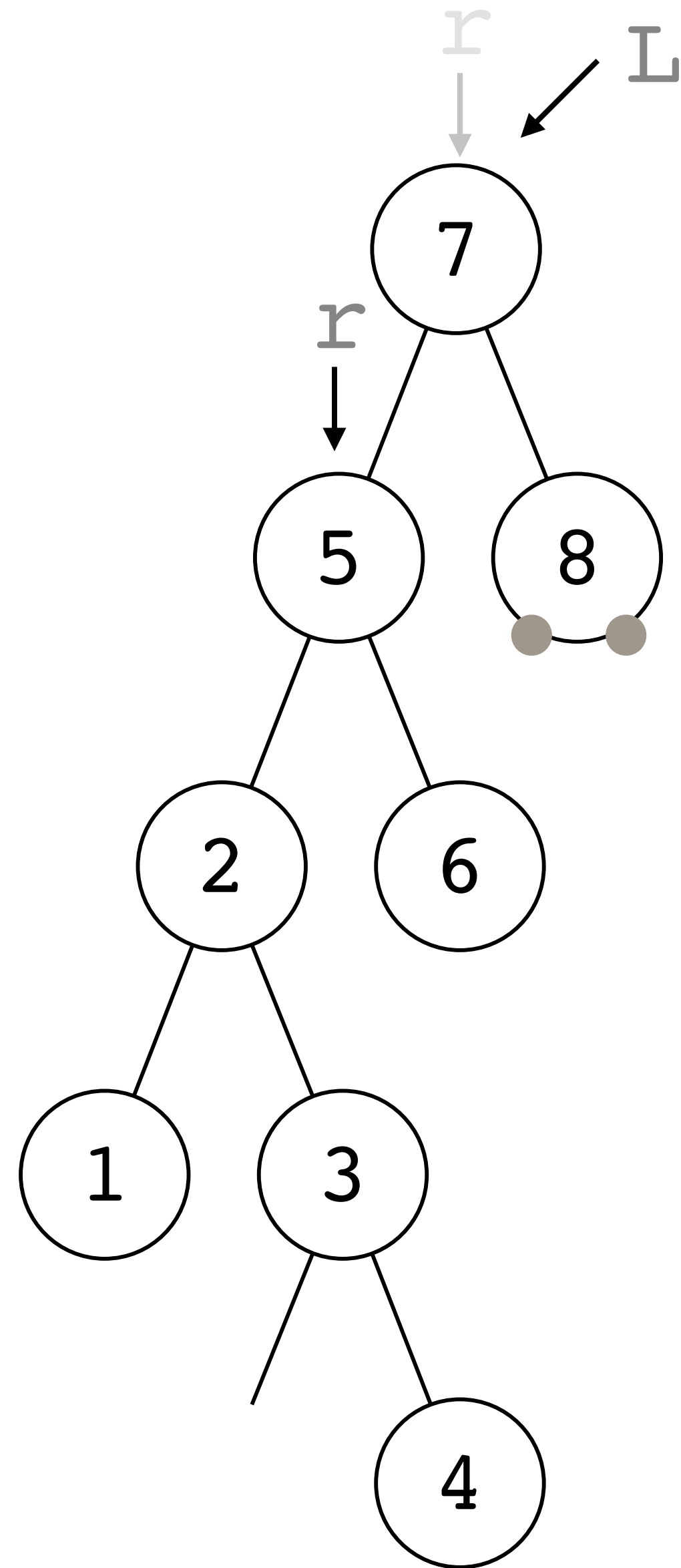
```
  linéariser(r=8, L=∅, n=0)
```

```
    8.droit ← ∅,  L ← 8,  n ← 1
```

```
    8.gauche ← ∅
```

```
  7.droit ← 8,  L ← 7,  n ← 2
```

```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L,  L ← r,  n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```



```
linéariser(r=7, L=∅, n=0)
```

```
linéariser(r=8, L=∅, n=0)
```

```
8.droit ← ∅, L ← 8, n ← 1
```

```
8.gauche ← ∅
```

```
7.droit ← 8, L ← 7, n ← 2
```

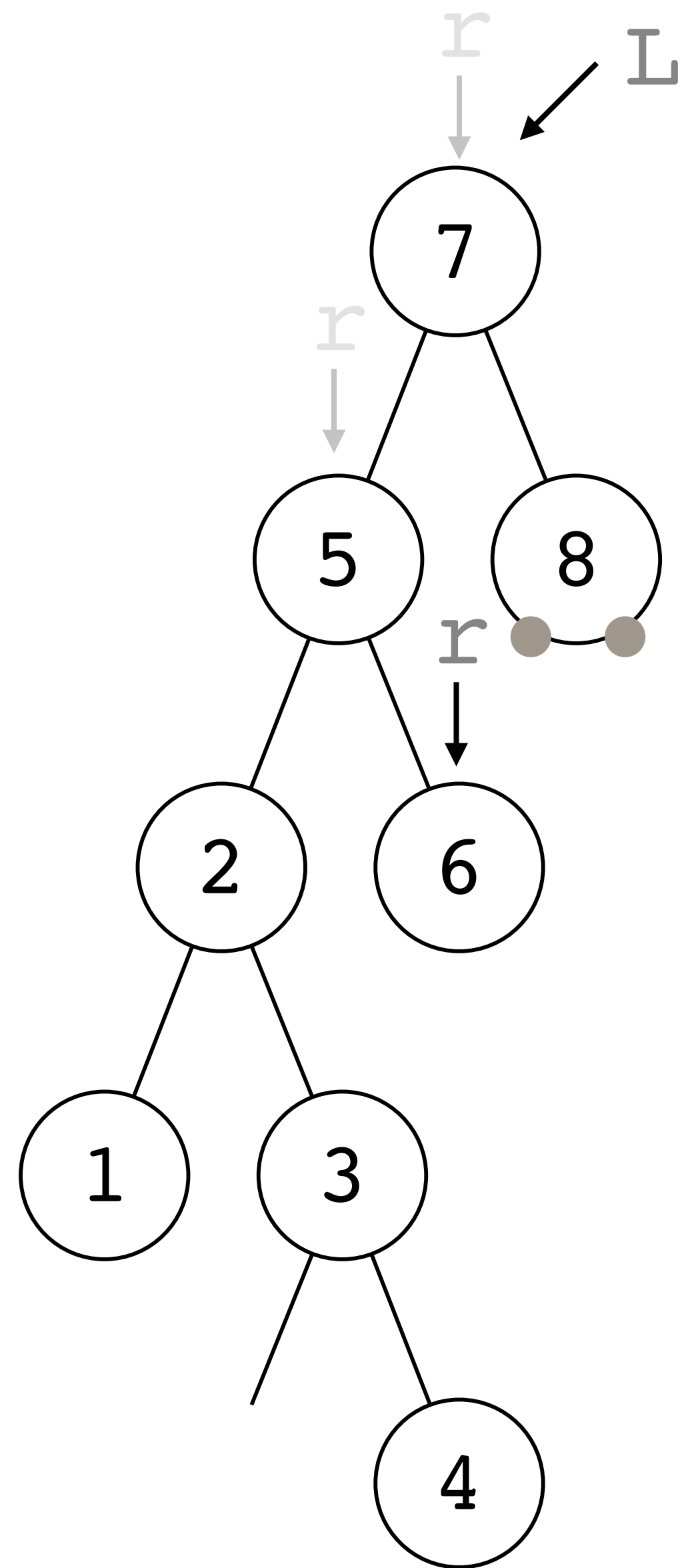
```
linéariser(r=5, L=7, n=2)
```

```
linéariser(r=6, L=7, n=2)
```

```
6.droit ← 7, L ← 6, n ← 3
```

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
  
```



```
linéariser(r=7, L=∅, n=0)
```

```
linéariser(r=8, L=∅, n=0)
```

```
8.droit ← ∅, L ← 8, n ← 1
```

```
8.gauche ← ∅
```

```
7.droit ← 8, L ← 7, n ← 2
```

```
linéariser(r=5, L=7, n=2)
```

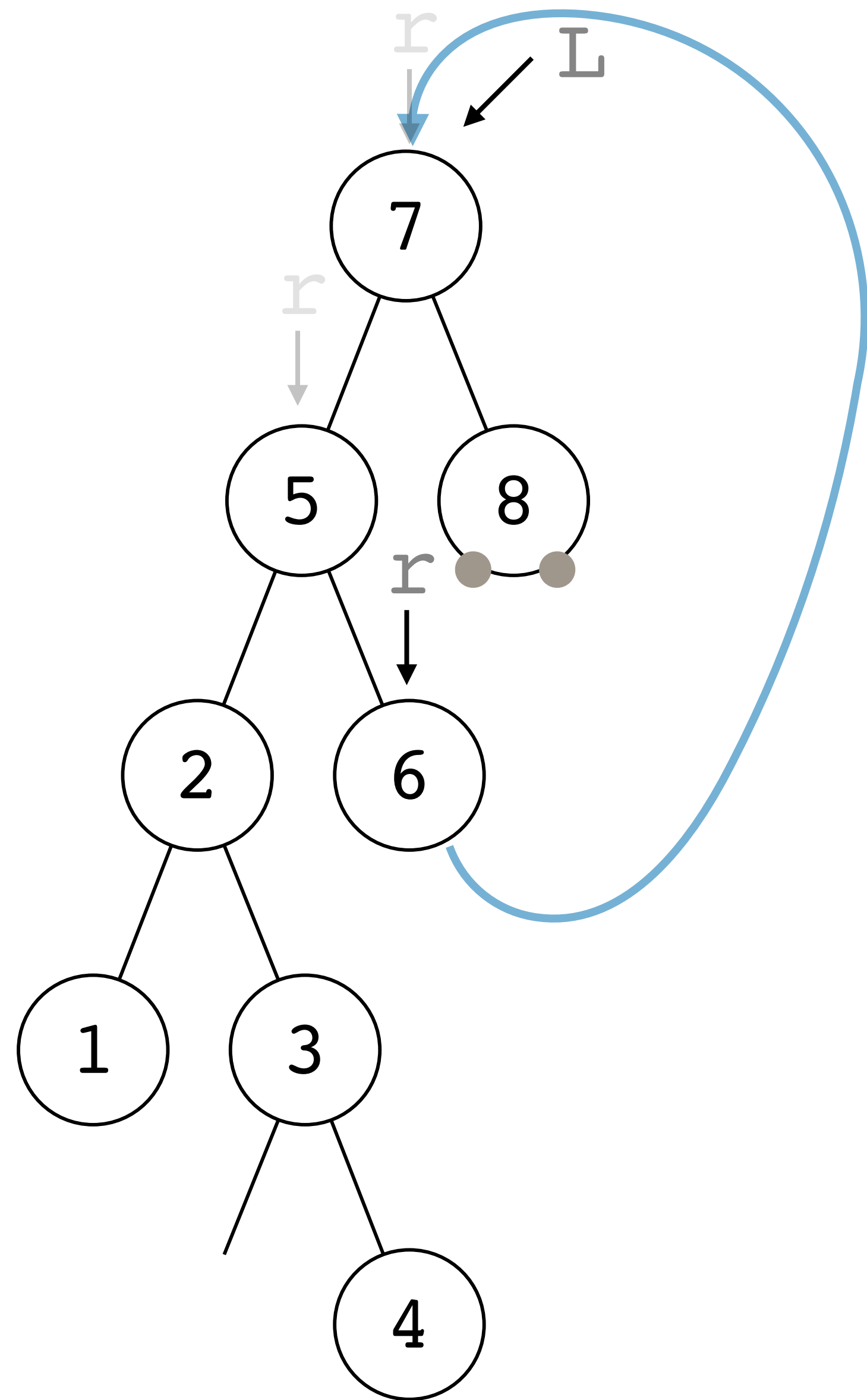
```
linéariser(r=6, L=7, n=2)
```

```
6.droit ← 7, L ← 6, n ← 3
```

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅

```



```
linéariser(r=7, L=∅, n=0)
```

```
linéariser(r=8, L=∅, n=0)
```

```
8.droit ← ∅, L ← 8, n ← 1
```

```
8.gauche ← ∅
```

```
7.droit ← 8, L ← 7, n ← 2
```

```
linéariser(r=5, L=7, n=2)
```

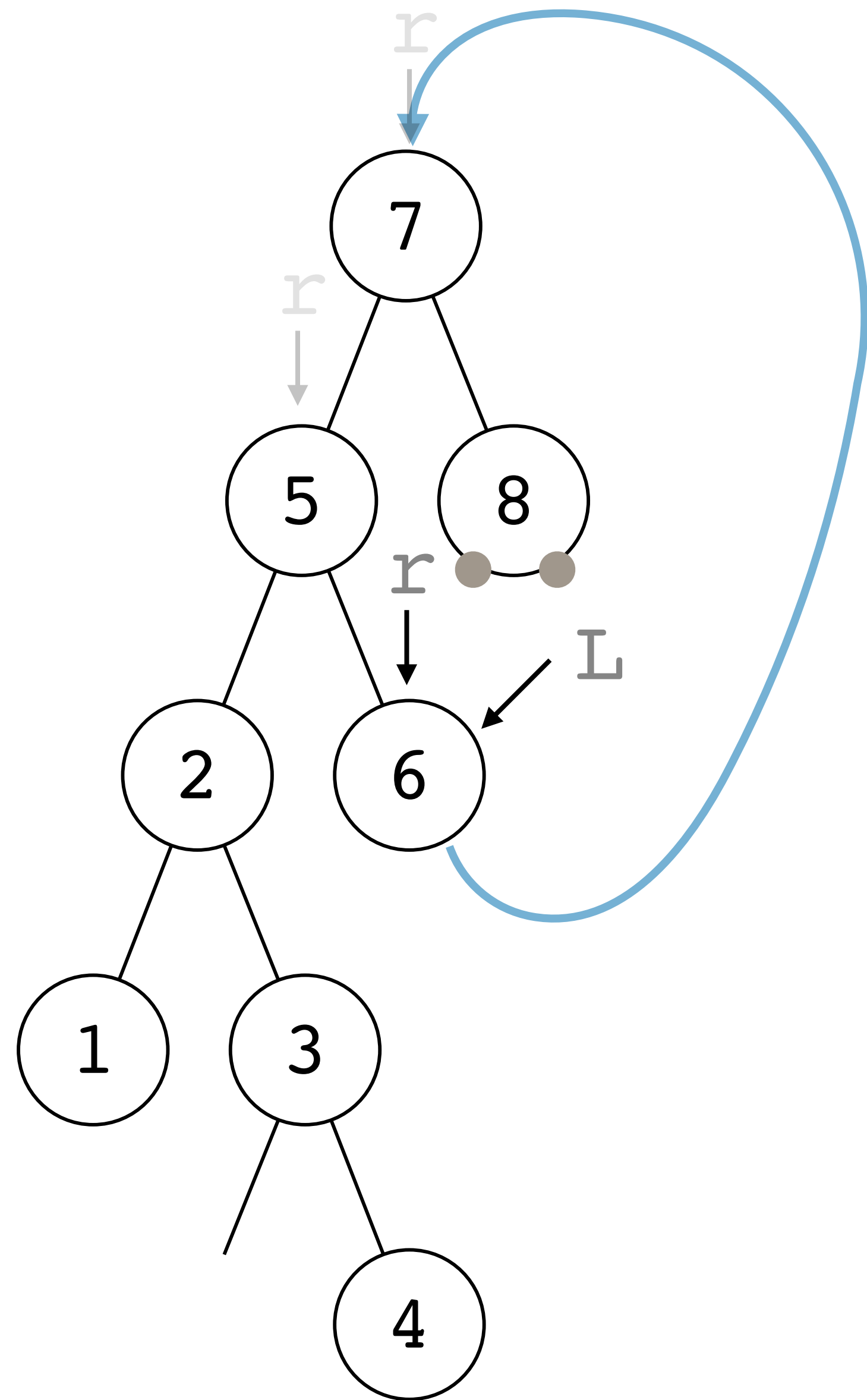
```
linéariser(r=6, L=7, n=2)
```

```
6.droit ← 7, L ← 6, n ← 3
```

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅

```



```
linéariser(r=7, L=∅, n=0)
```

```
linéariser(r=8, L=∅, n=0)
```

```
8.droit ← ∅, L ← 8, n ← 1
```

```
8.gauche ← ∅
```

```
7.droit ← 8, L ← 7, n ← 2
```

```
linéariser(r=5, L=7, n=2)
```

```
linéariser(r=6, L=7, n=2)
```

```
6.droit ← 7, L ← 6, n ← 3
```

```
fonction linéariser (r, ref L, ref n)
```

```
si r != ∅,
```

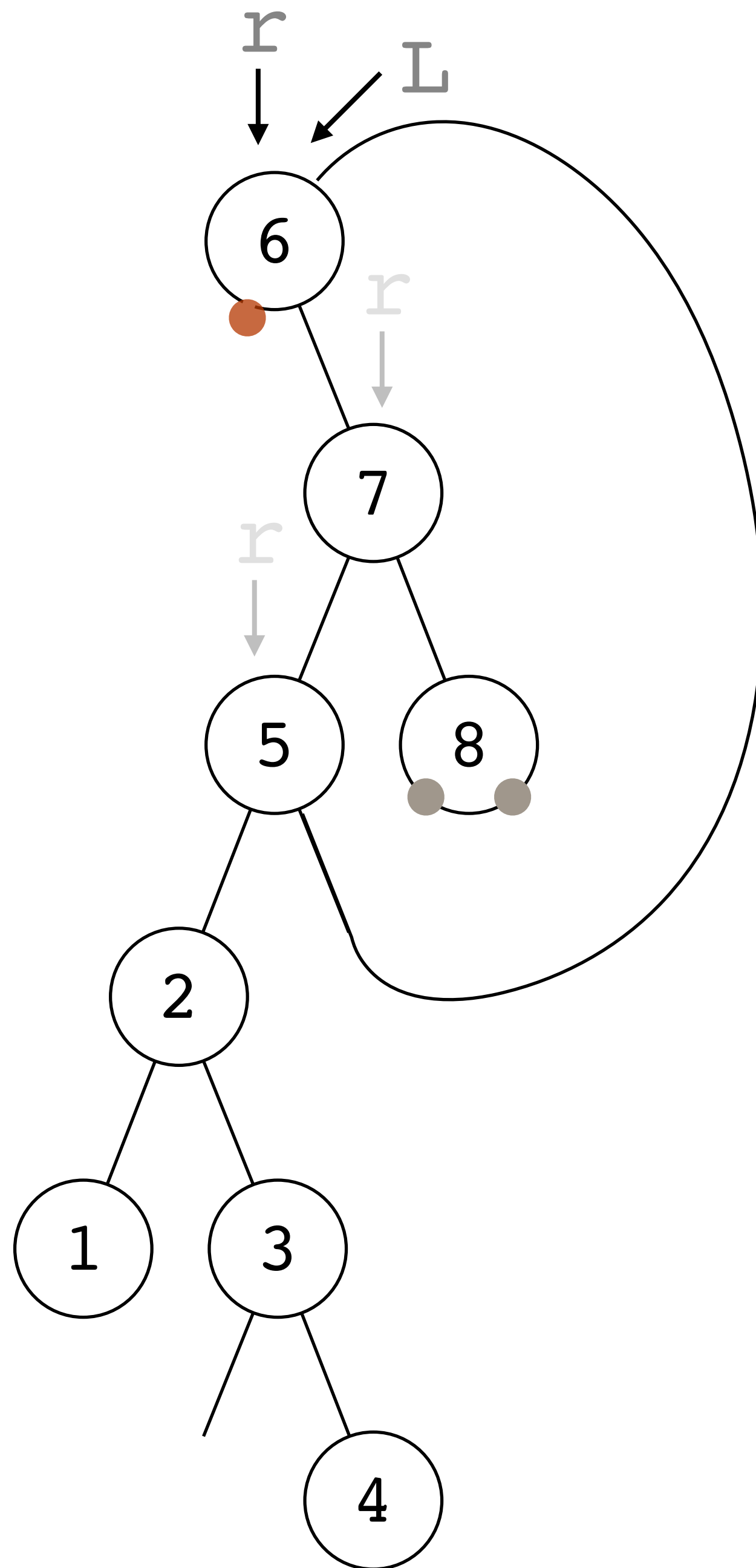
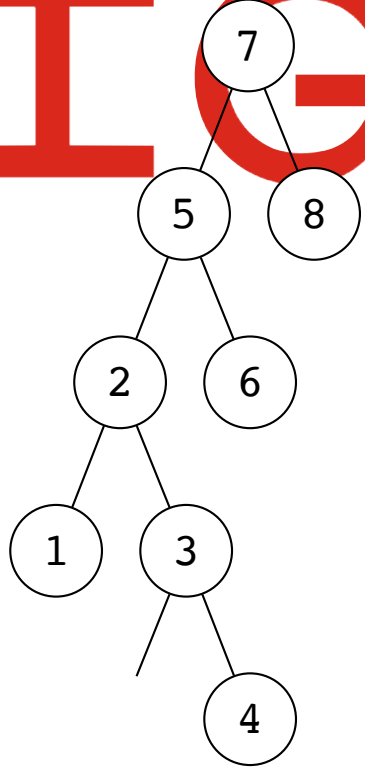
```
linéariser(r.droit, L, n)
```

```
r.droit ← L, L ← r, n ← n + 1
```

```
linéariser(r.gauche, L, n)
```

```
r.gauche ← ∅
```

HEIG^{VD}



linéariser($r=8$, $L=\emptyset$, $n=0$)

$8.\text{droit} \leftarrow \emptyset$, $L \leftarrow 8$, $n \leftarrow 1$

$7.\text{droit} \leftarrow 8$, $L \leftarrow 7$, $n \leftarrow 2$

linéariser($r=5$, $L=7$, $n=2$)

linéariser($r=6$, $L=7$, $n=2$)

$6.\text{droit} \leftarrow 7$, $L \leftarrow 6$, $n \leftarrow 3$

$6.\text{gauche} \leftarrow \emptyset$

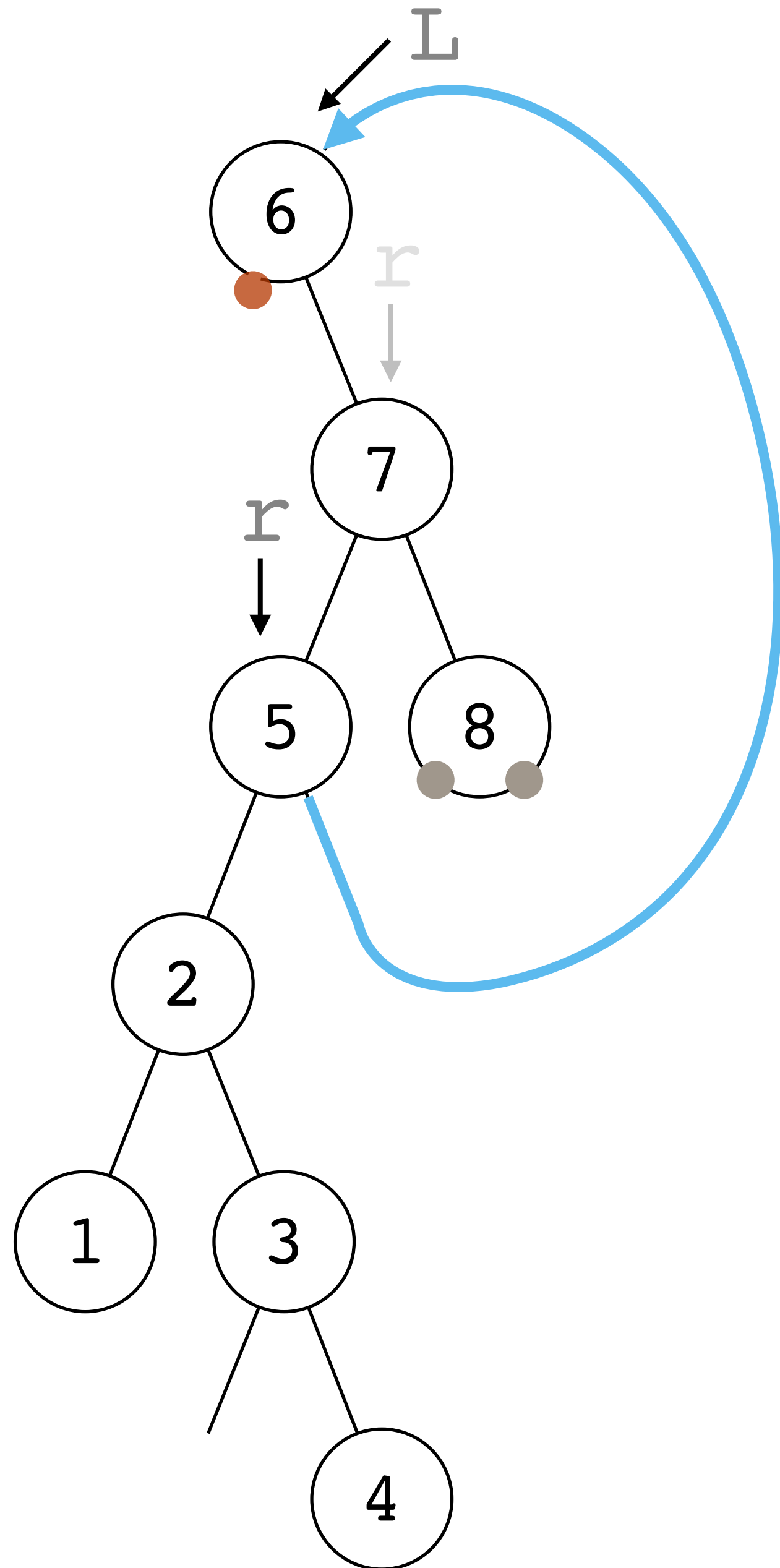
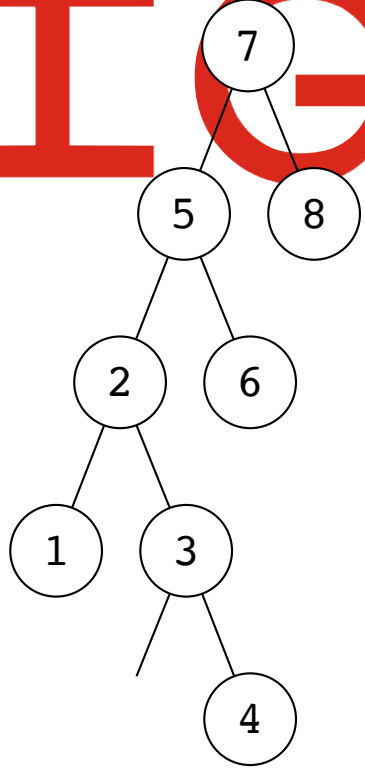
$5.\text{droit} \leftarrow 6$, $L \leftarrow 5$, $n \leftarrow 4$

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
  
```



HEIG^{VD}



linéariser($r=8$, $L=\emptyset$, $n=0$)

$8.\text{droit} \leftarrow \emptyset$, $L \leftarrow 8$, $n \leftarrow 1$

$7.\text{droit} \leftarrow 8$, $L \leftarrow 7$, $n \leftarrow 2$

linéariser($r=5$, $L=7$, $n=2$)

linéariser($r=6$, $L=7$, $n=2$)

$6.\text{droit} \leftarrow 7$, $L \leftarrow 6$, $n \leftarrow 3$

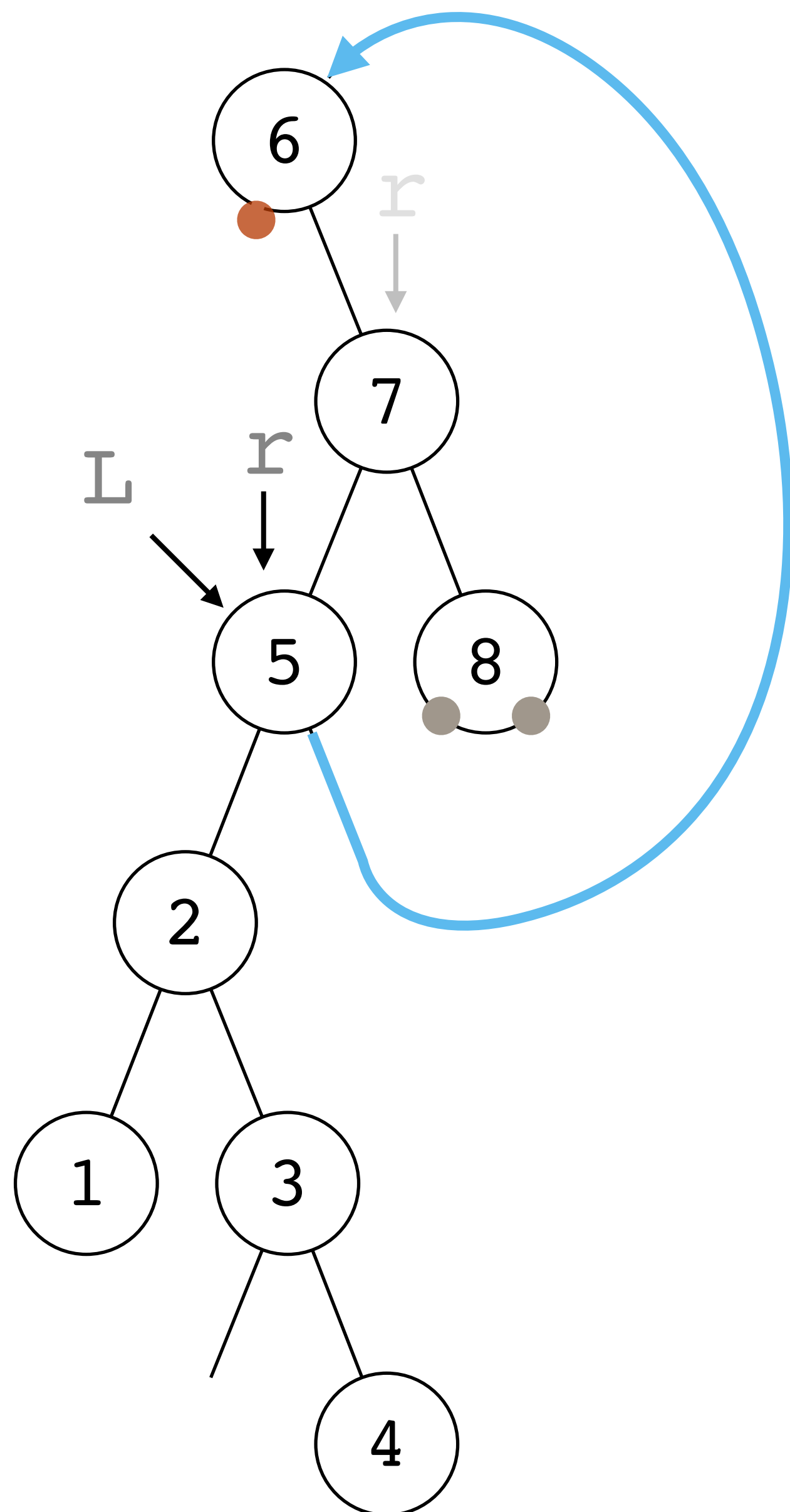
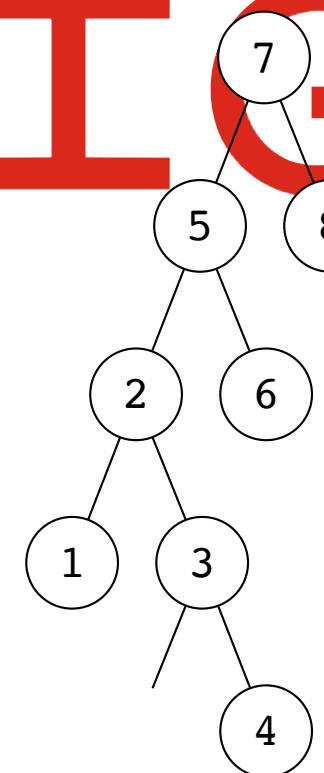
$6.\text{gauche} \leftarrow \emptyset$

$5.\text{droit} \leftarrow 6$, $L \leftarrow 5$, $n \leftarrow 4$



```
fonction linéariser (r, ref L, ref n)
si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```

HEIG^{VD}



linéariser($r=8$, $L=\emptyset$, $n=0$)

8.droit $\leftarrow \emptyset$, $L \leftarrow 8$, $n \leftarrow 1$

7.droit $\leftarrow 8$, $L \leftarrow 7$, $n \leftarrow 2$

linéariser($r=5$, $L=7$, $n=2$)

linéariser($r=6$, $L=7$, $n=2$)

6.droit $\leftarrow 7$, $L \leftarrow 6$, $n \leftarrow 3$

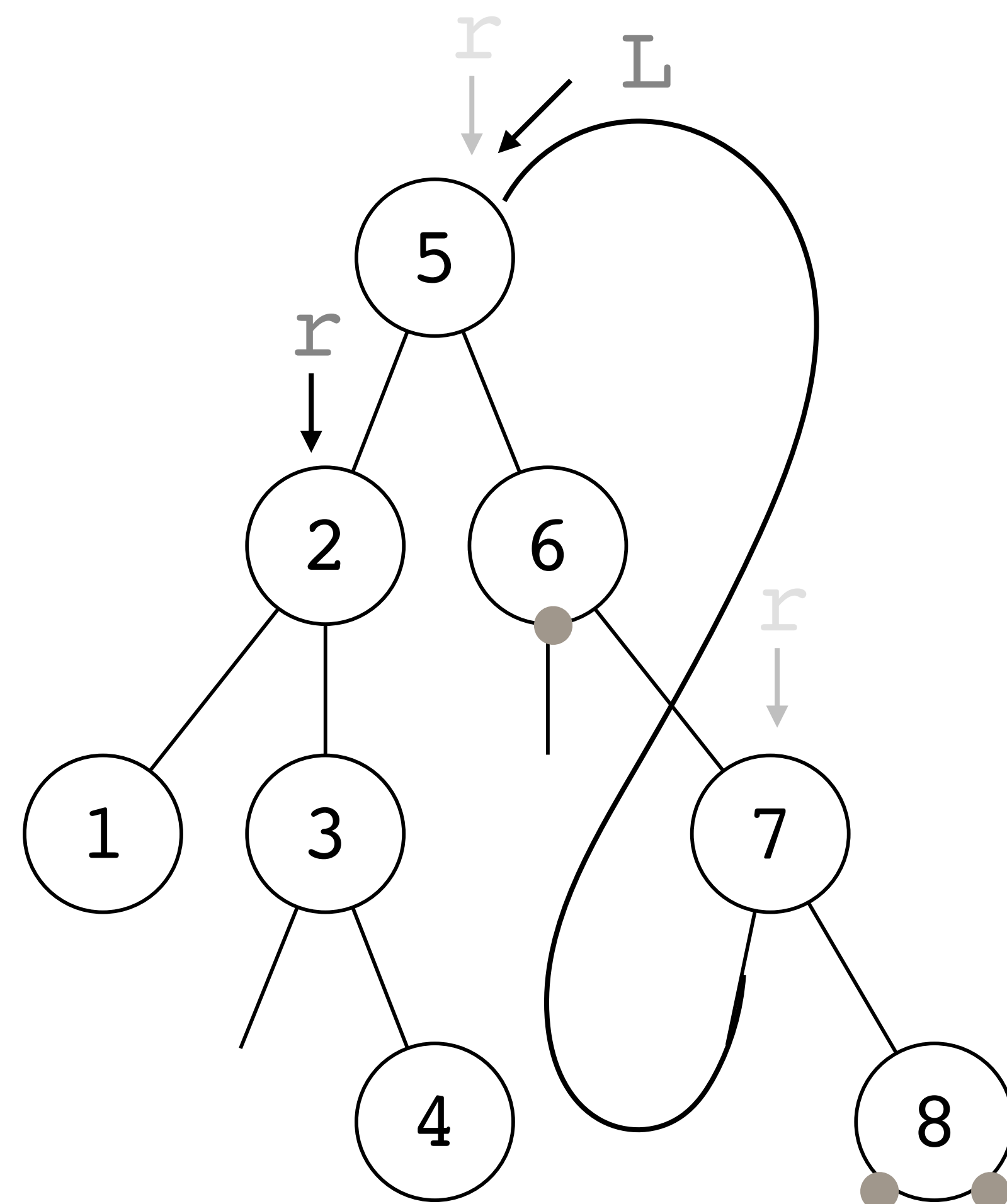
6.gauche $\leftarrow \emptyset$

5.droit $\leftarrow 6$, $L \leftarrow 5$, $n \leftarrow 4$



```
fonction linéariser (r, ref L, ref n)
si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```

HEIG^{VD}



```
linéariser(r=5, L=7, n=2)
```

```
linéariser(r=6, L=7, n=2)
```

```
6.droit ← 7, L ← 6, n ← 3
```

```
6.gauche ← ∅
```

```
5.droit ← 6, L ← 5, n ← 4
```

```
linéariser(r=2, L=5, n=4)
```

```
linéariser(r=3, L=5, n=4)
```

```
linéariser(r=4, L=5, n=4)
```

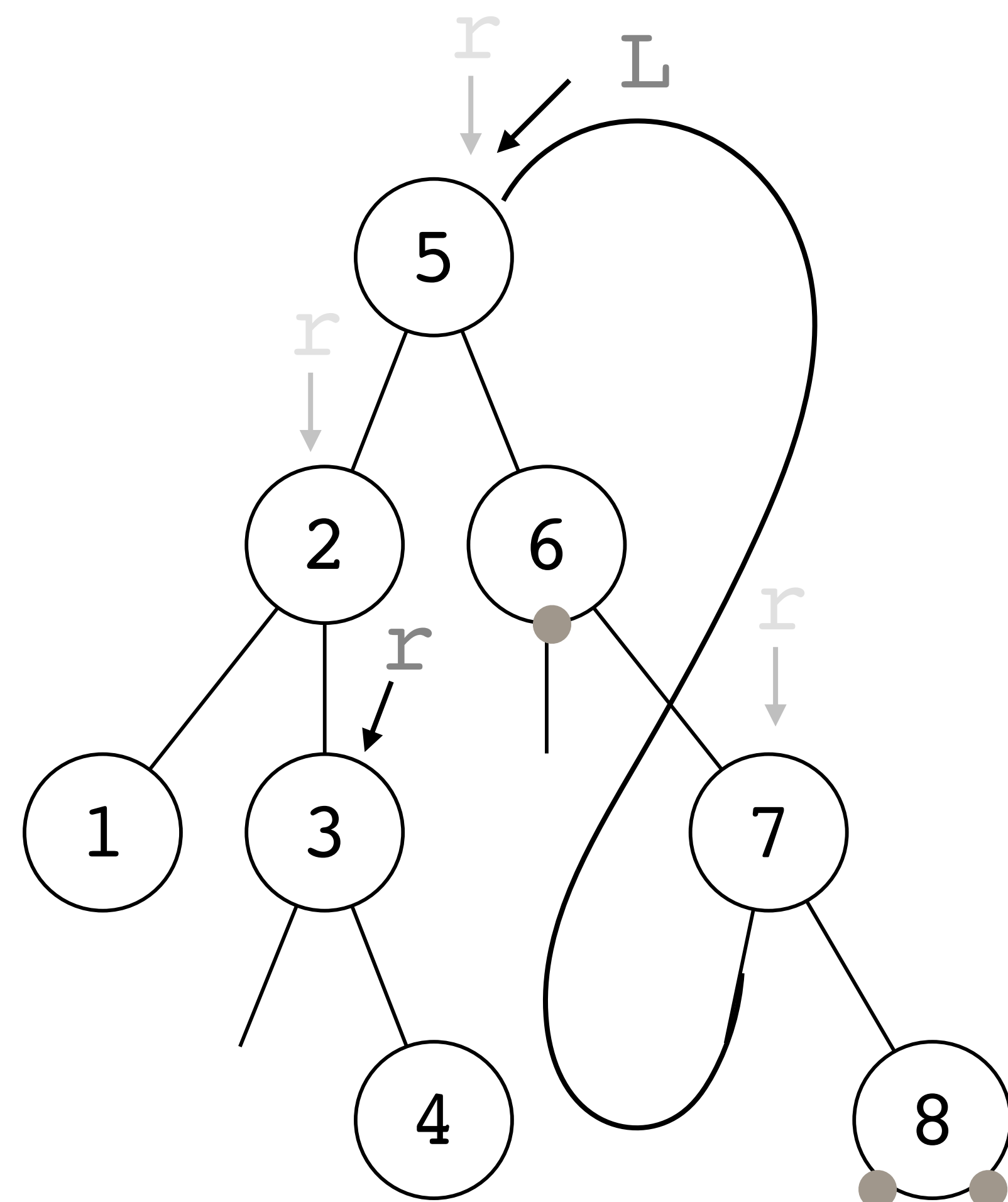
```
4.droit ← 5, L ← 4, n ← 5
```

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅

```

HEIG^{VD}



7.droit ← 6, L ← 7, n ← 2
linéariser(r=5, L=7, n=2)

linéariser(r=6, L=7, n=2)

6.droit ← 7, L ← 6, n ← 3

6.gauche ← ∅

5.droit ← 6, L ← 5, n ← 4

linéariser(r=2, L=5, n=4)

linéariser(r=3, L=5, n=4)

linéariser(r=4, L=5, n=4)

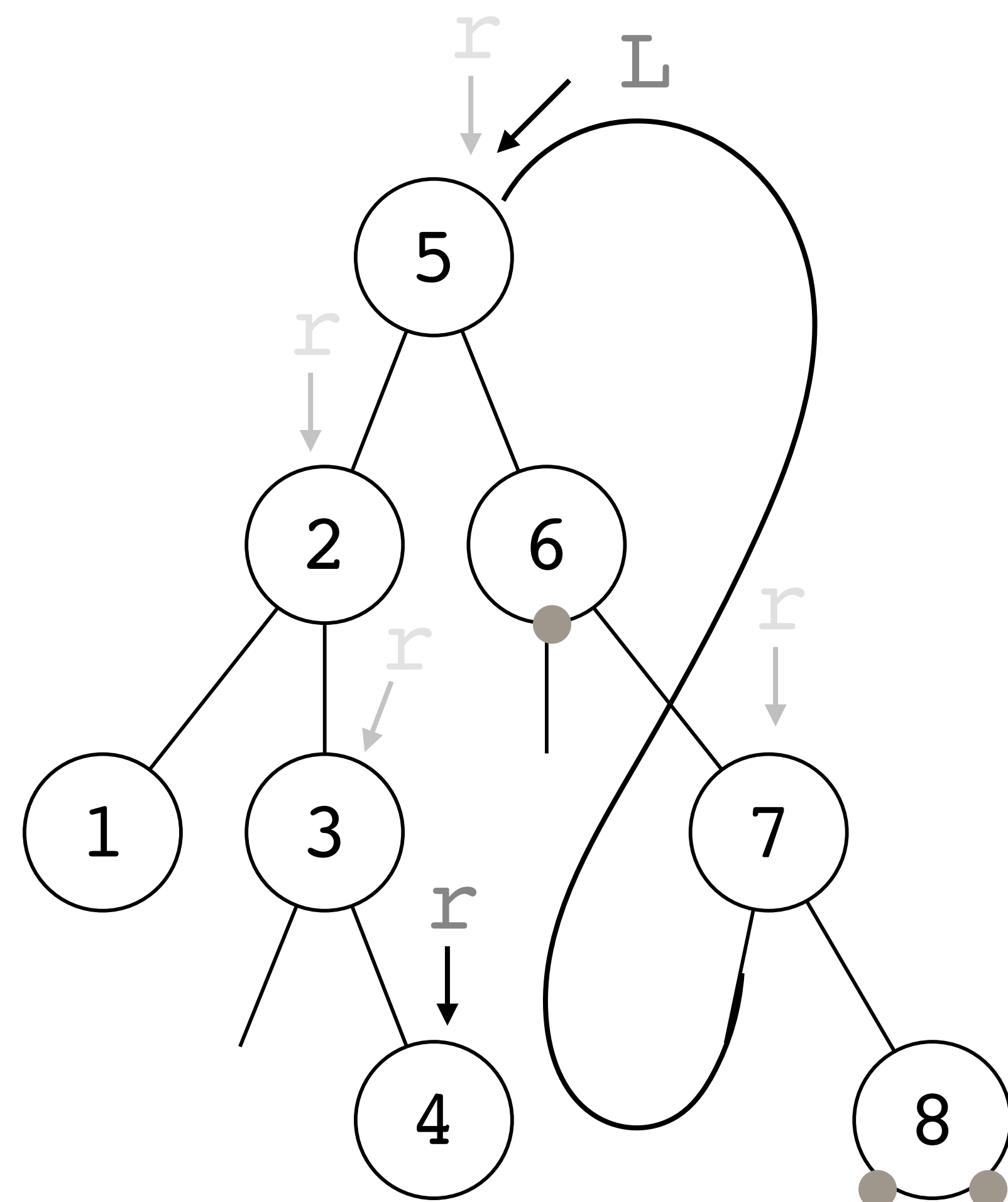
4.droit ← 5, L ← 4, n ← 5

```
fonction linéariser (r, ref L, ref n)
si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```



HEIG

VD



`linéariser(r=5, L=7, n=2)`

`linéariser(r=6, L=7, n=2)`

`6.droit ← 7, L ← 6, n ← 3`

`6.gauche ← ∅`

`5.droit ← 6, L ← 5, n ← 4`

`linéariser(r=2, L=5, n=4)`

`linéariser(r=3, L=5, n=4)`

`linéariser(r=4, L=5, n=4)`

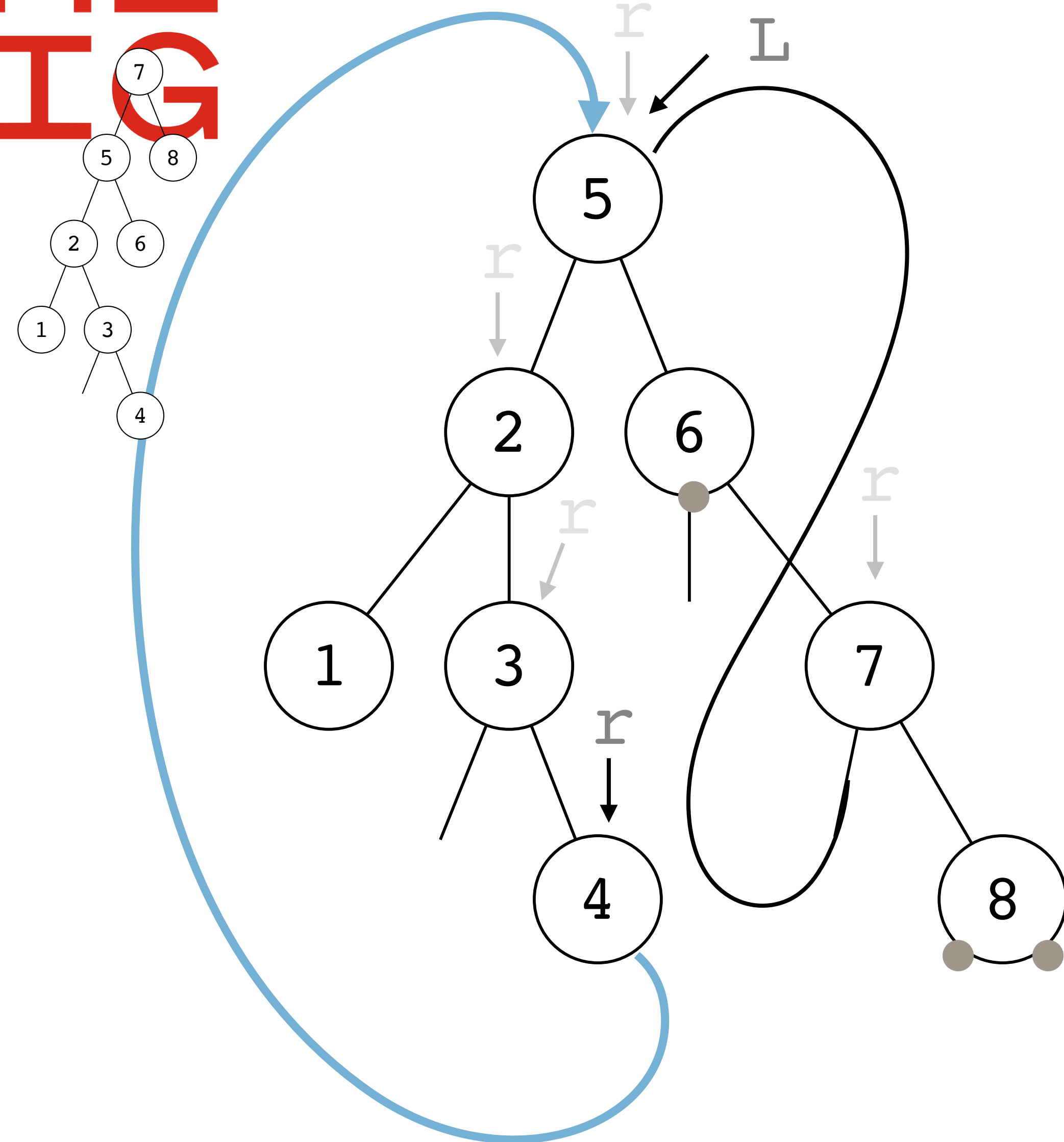
`4.droit ← 5, L ← 4, n ← 5`

```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
  
```



HEIG^{VD}



```
linéariser(r=5, L=7, n=2)
```

```
linéariser(r=6, L=7, n=2)
```

```
6.droit ← 7, L ← 6, n ← 3
```

```
6.gauche ← ∅
```

```
5.droit ← 6, L ← 5, n ← 4
```

```
linéariser(r=2, L=5, n=4)
```

```
linéariser(r=3, L=5, n=4)
```

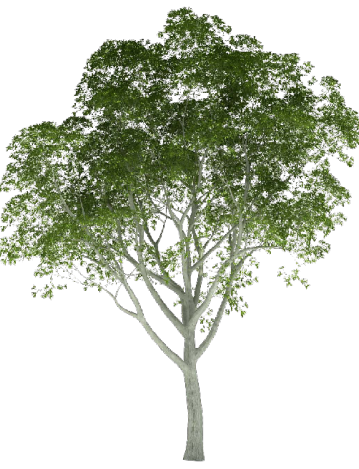
```
linéariser(r=4, L=5, n=4)
```

```
4.droit ← 5, L ← 4, n ← 5
```

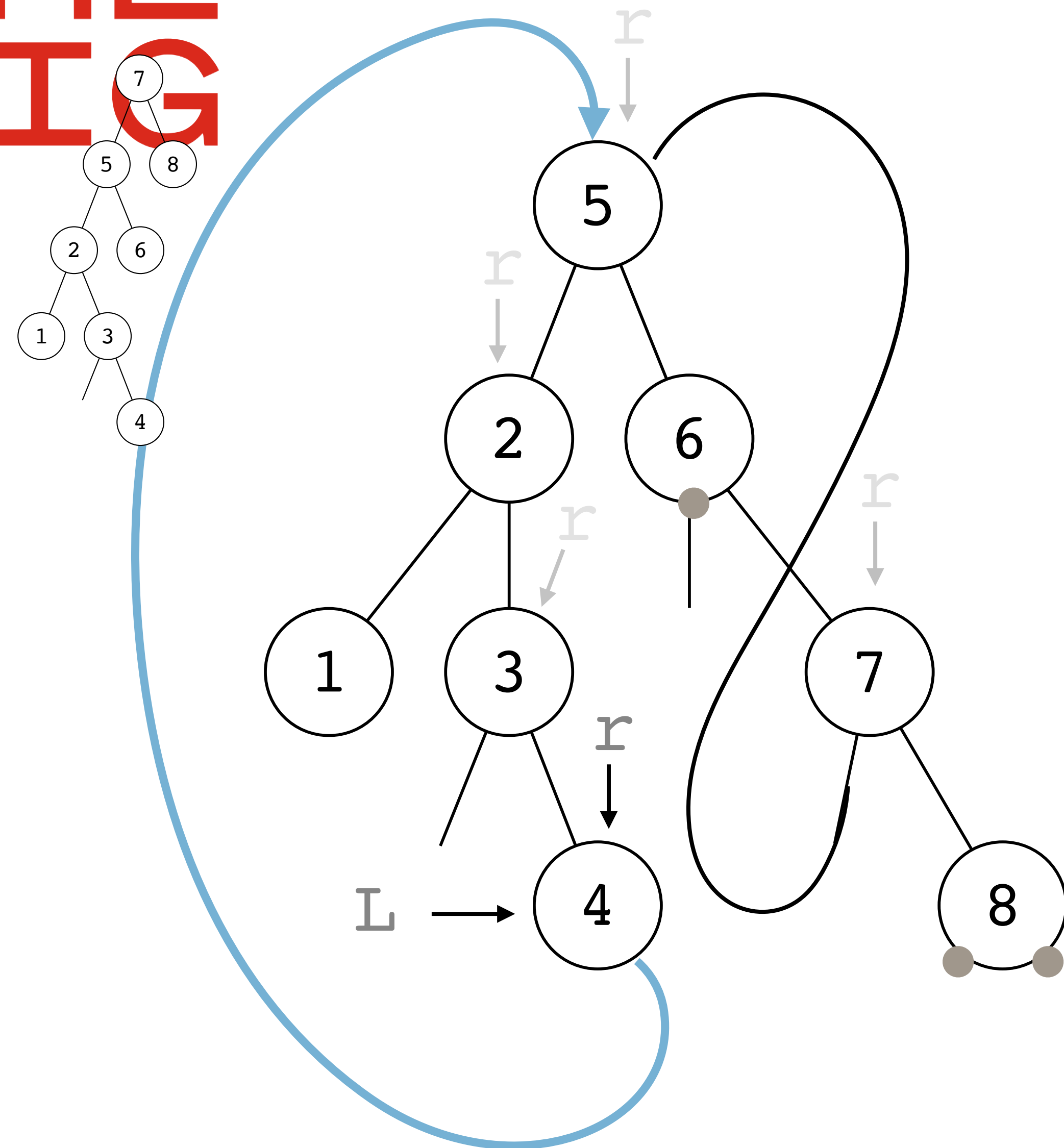
```

fonction linéariser (r, ref L, ref n)
  si r ≠ ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅

```



HEIG^{VD}



```
linéariser(r=5, L=7, n=2)
```

```
linéariser(r=6, L=7, n=2)
```

```
6.droit ← 7, L ← 6, n ← 3
```

```
6.gauche ← ∅
```

```
5.droit ← 6, L ← 5, n ← 4
```

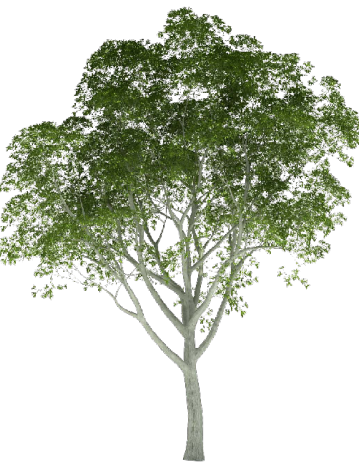
```
linéariser(r=2, L=5, n=4)
```

```
linéariser(r=3, L=5, n=4)
```

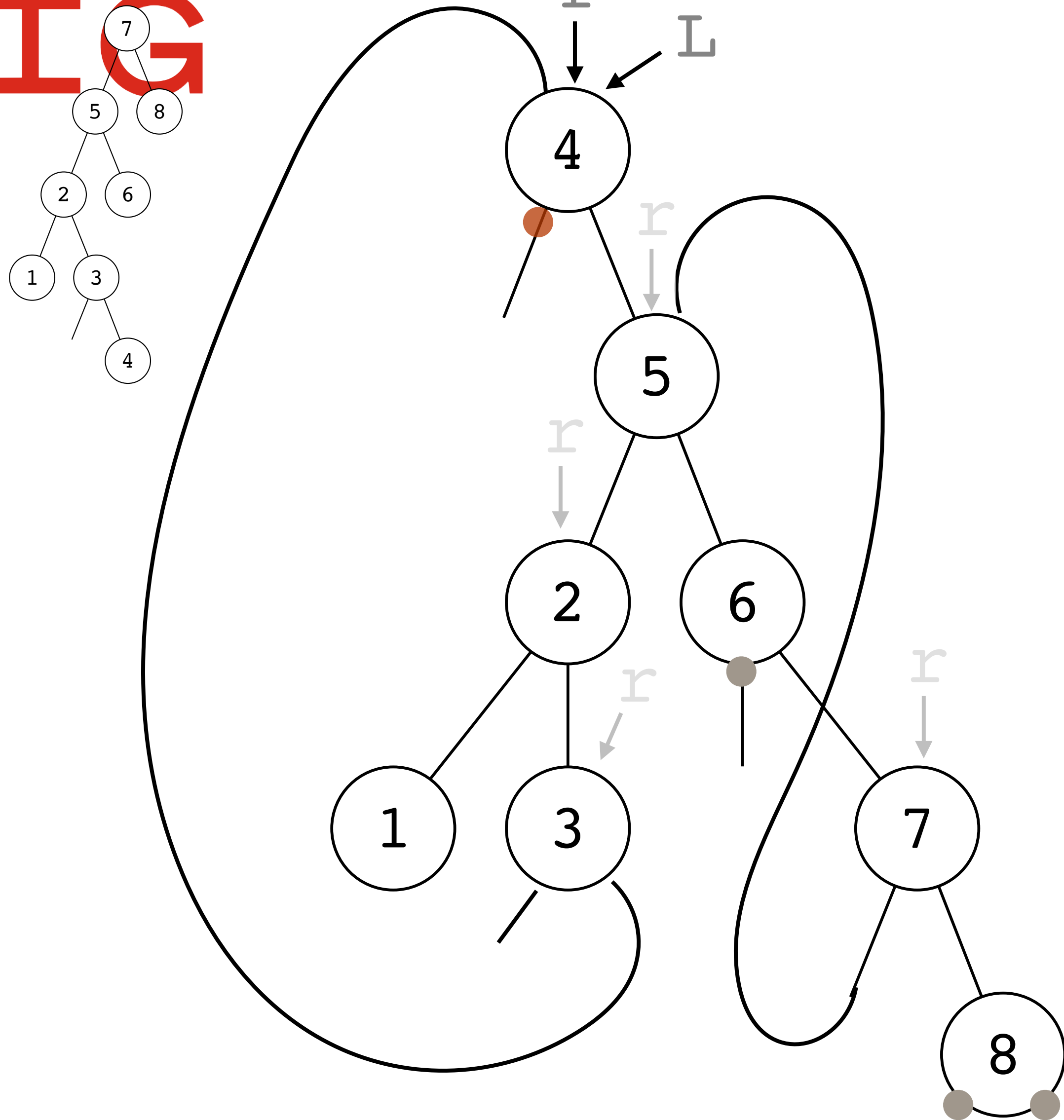
```
linéariser(r=4, L=5, n=4)
```

```
4.droit ← 5, L ← 4, n ← 5
```

```
fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L, L ← r, n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
```



HEIG^{VD}



6.droit $\leftarrow$ 7, L $\leftarrow$ 6, n $\leftarrow$ 3

6.gauche $\leftarrow$ $\emptyset$

5.droit $\leftarrow$ 6, L $\leftarrow$ 5, n $\leftarrow$ 4

linéariser(r=2, L=5, n=4)

linéariser(r=3, L=5, n=4)

linéariser(r=4, L=5, n=4)

4.droit $\leftarrow$ 5, L $\leftarrow$ 4, n $\leftarrow$ 5

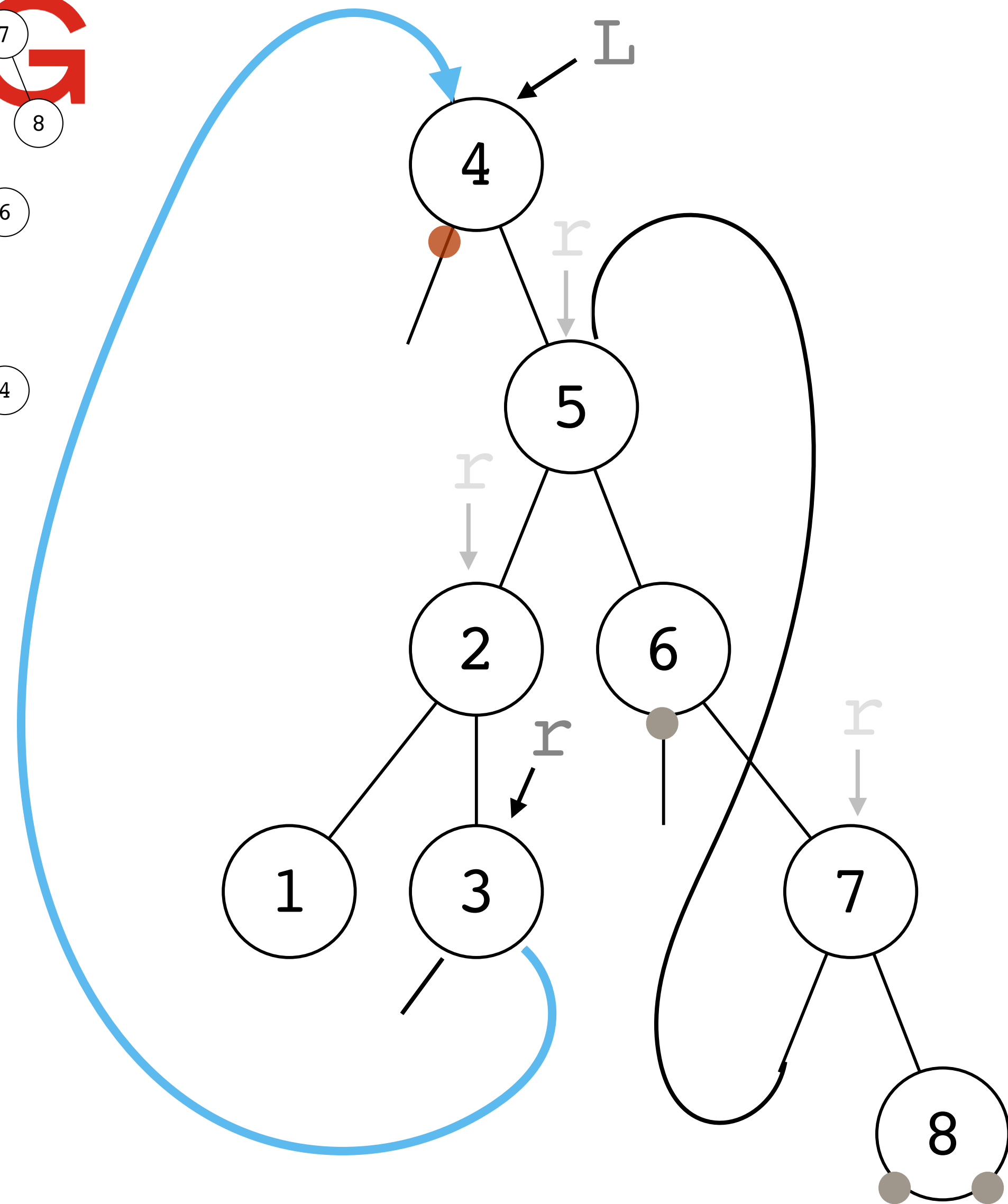
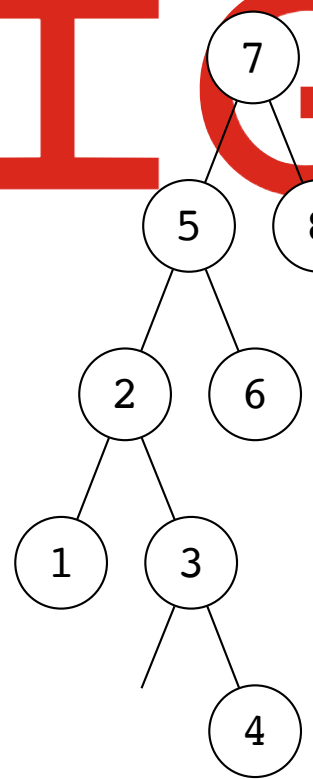
4.gauche $\leftarrow$ $\emptyset$

3.droit $\leftarrow$ 4, L $\leftarrow$ 3, n $\leftarrow$ 6



```

fonction linéariser (r, ref L, ref n)
  si r  $\neq$   $\emptyset$ ,
    linéariser(r.droit, L, n)
    r.droit  $\leftarrow$  L, L  $\leftarrow$  r, n  $\leftarrow$  n + 1
    linéariser(r.gauche, L, n)
    r.gauche  $\leftarrow$   $\emptyset$ 
  
```



A large, mature tree with a thick trunk and a dense, rounded canopy of green leaves. The tree is shown against a plain white background.

A large, mature tree with a thick trunk and a dense, rounded canopy of green leaves. The tree is shown against a plain white background.

A large, mature tree with a thick trunk and a dense, rounded canopy of green leaves. The tree is shown against a plain white background.

A large, mature tree with a thick trunk and a dense, rounded canopy of green leaves. The tree is shown against a plain white background.

A large, mature tree with a thick trunk and a dense, rounded canopy of green leaves. The tree is shown against a plain white background.

A large, mature tree with a thick trunk and a dense, rounded canopy of green leaves. The tree is shown against a plain white background.

A large, mature tree with a thick trunk and a dense, rounded canopy of green leaves. The tree is shown against a plain white background.

A large, mature tree with a thick trunk and a dense, rounded canopy of green leaves. The tree is shown against a plain white background.

A large, mature tree with a thick trunk and a dense, rounded canopy of green leaves. The tree is shown against a plain white background.

```
fonction linéariser (r, ref L, ref n)
```

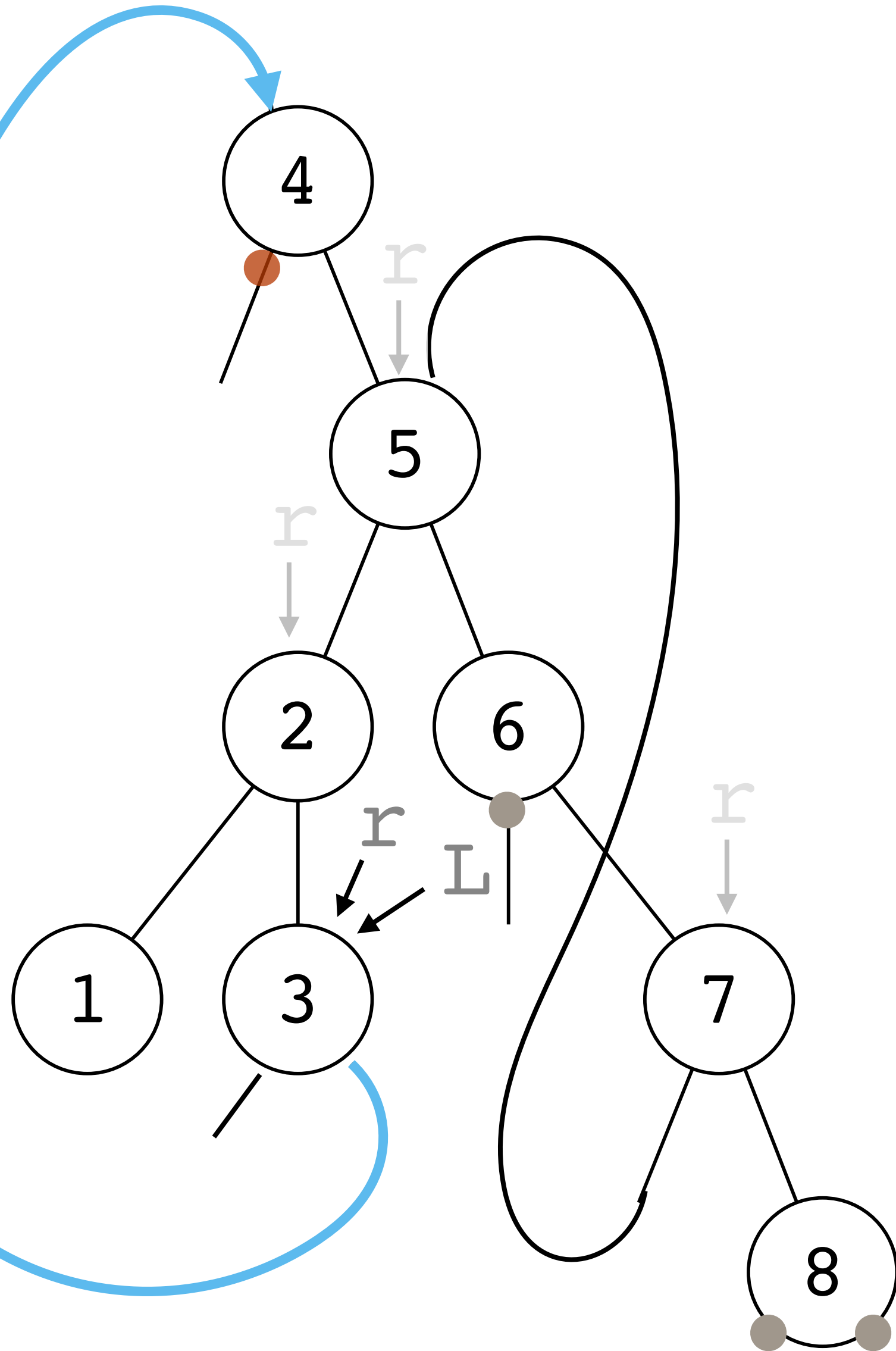
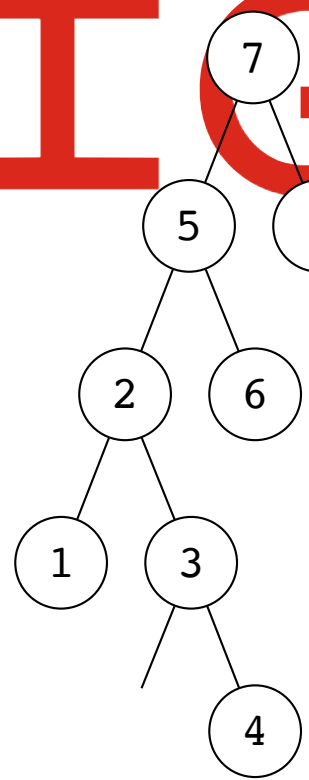
```
fonction linéariser (r, ref L, ref n)
```

```
fonction linéariser (r, ref L, ref n)
```

```
fonction linéariser (r, ref L, ref n)
```

```
fonction linéariser (r, ref L, ref n)
```

```
fonction linéariser (r, ref L, ref n)
```



6.gauche $\leftarrow \emptyset$

5.droit $\leftarrow$ 6, L $\leftarrow$ 5, n $\leftarrow$ 4

linéariser(r=2, L=5, n=4)

linéariser(r=3, L=5, n=4)

```
linéariser(r=4, L=5, n=4)
```

4.droit $\leftarrow 5$, L $\leftarrow 4$, n $\leftarrow 5$

4. gauche $\leftarrow \emptyset$

3. `droit` $\leftarrow$ 4, `L` $\leftarrow$ 3, `n` $\leftarrow$ 6

```
fonction linéariser (r, ref L, ref n)
```

si $r \neq \emptyset$,

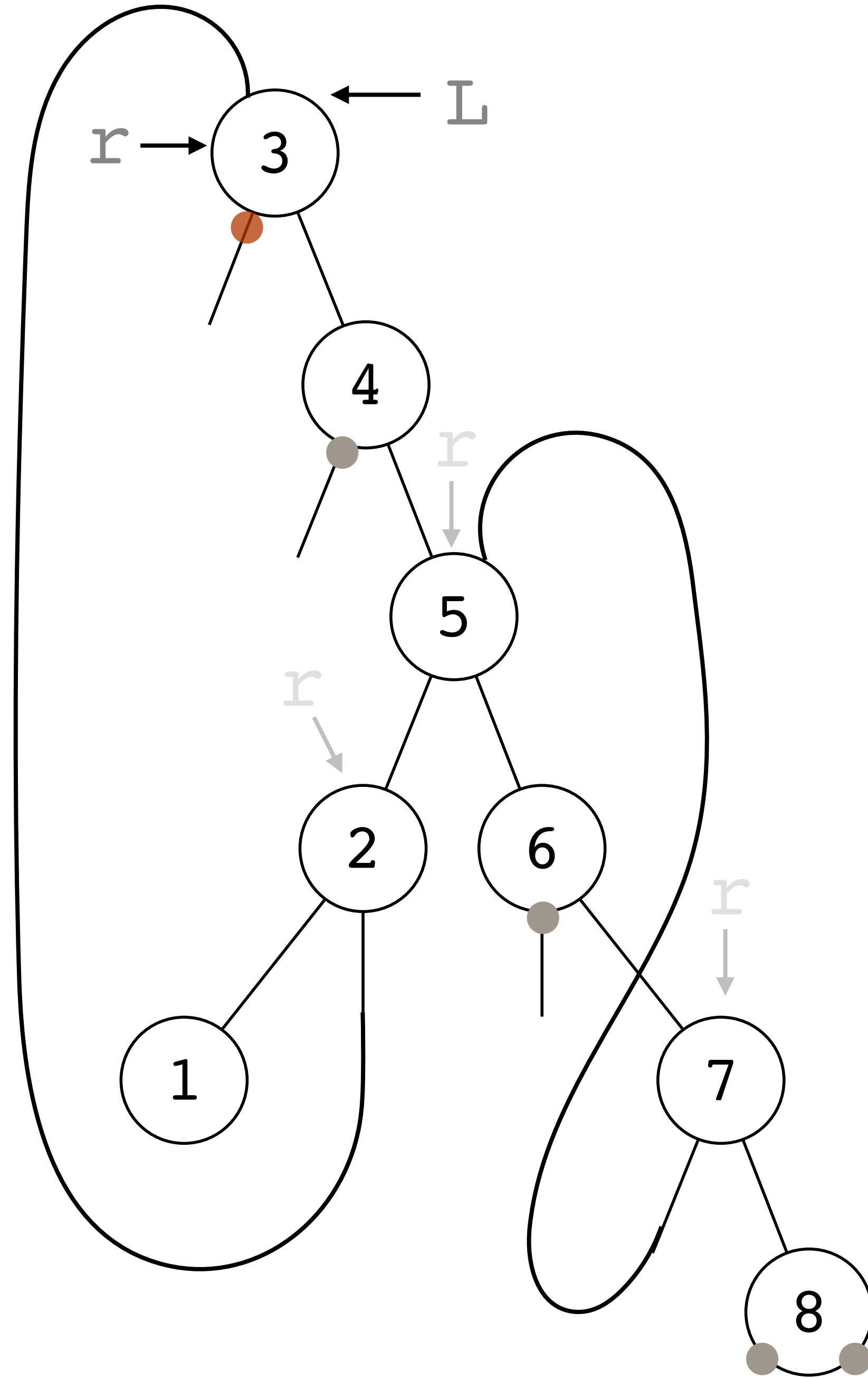
```
linéariser(r.droit, L, n)
```

```
r.droit ← L,  L ← r,  n ← n + 1
```

```
linéariser(r.gauche, L, n)
```

```
r.gauche ← ∅
```





2. **droit** $\leftarrow 3$, **L** $\leftarrow 2$, **n** $\leftarrow 7$



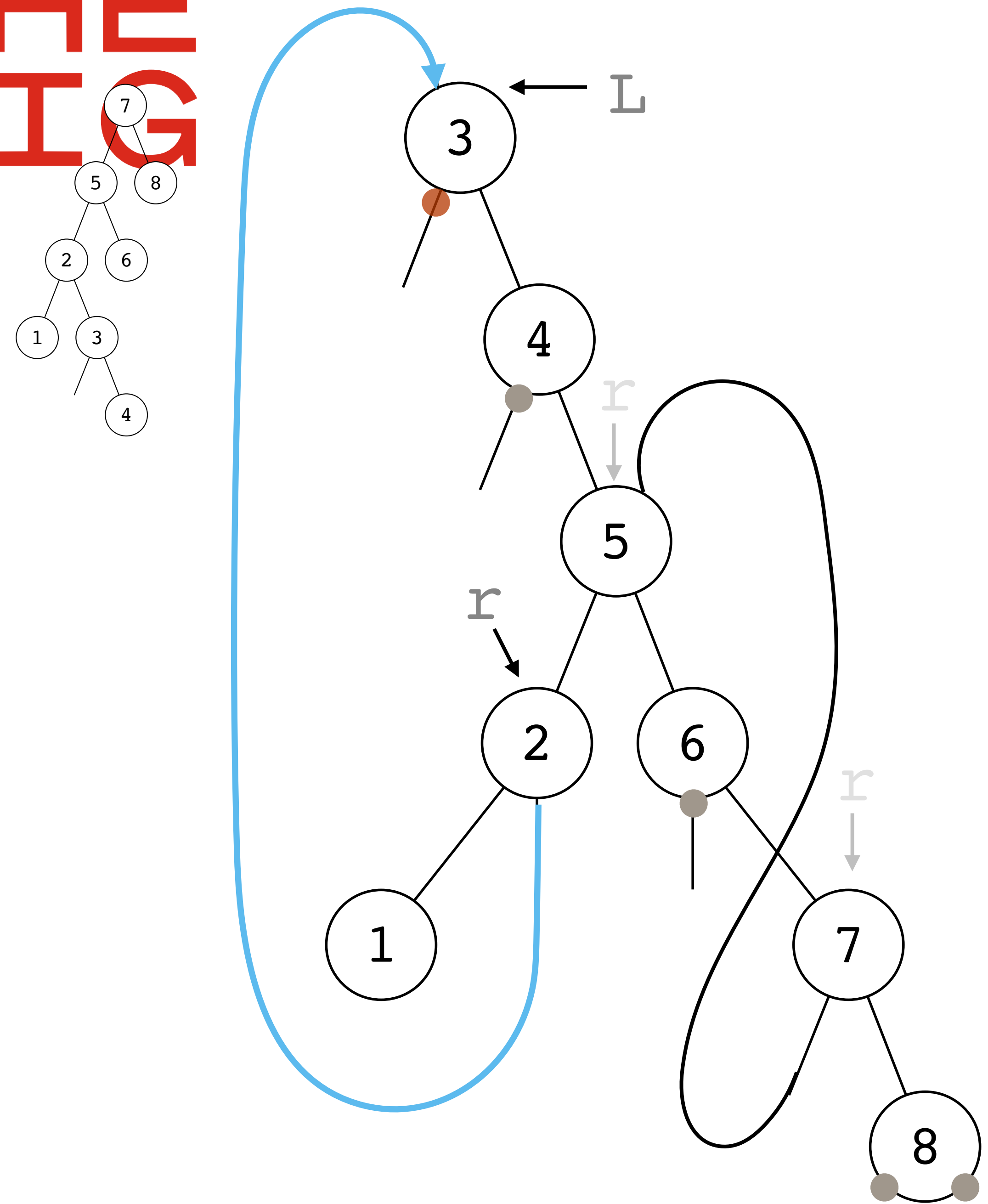
```

fonction linéariser (r, ref L, ref n)
    si r != ∅,
        linéariser(r.droit, L, n)
        r.droit ← L,    L ← r,    n ← n + 1
        linéariser(r.gauche, L, n)
        r.gauche ← ∅

```

```

graph TD
    5((5)) --- 2((2))
    5 --- 6((6))
    2 --- 1((1))
    2 --- 3((3))
    3 --- 4((4))
    7((7)) --- 5
    7 --- 8((8))
  
```



```
5.droit ← 6,   L ← 5,   n ← 4
linéariser(r=2, L=5, n=4)
    linéariser(r=3, L=5, n=4)
```



```

linéariser(r=4, L=5, n=4)
    4.droit ← 5,    L ← 4,    n ← 5
    4.gauche ← ∅
    3.droit ← 4,    L ← 3,    n ← 6
    3.gauche ← ∅
    2.droit ← 3,    L ← 2,    n ← 7

```

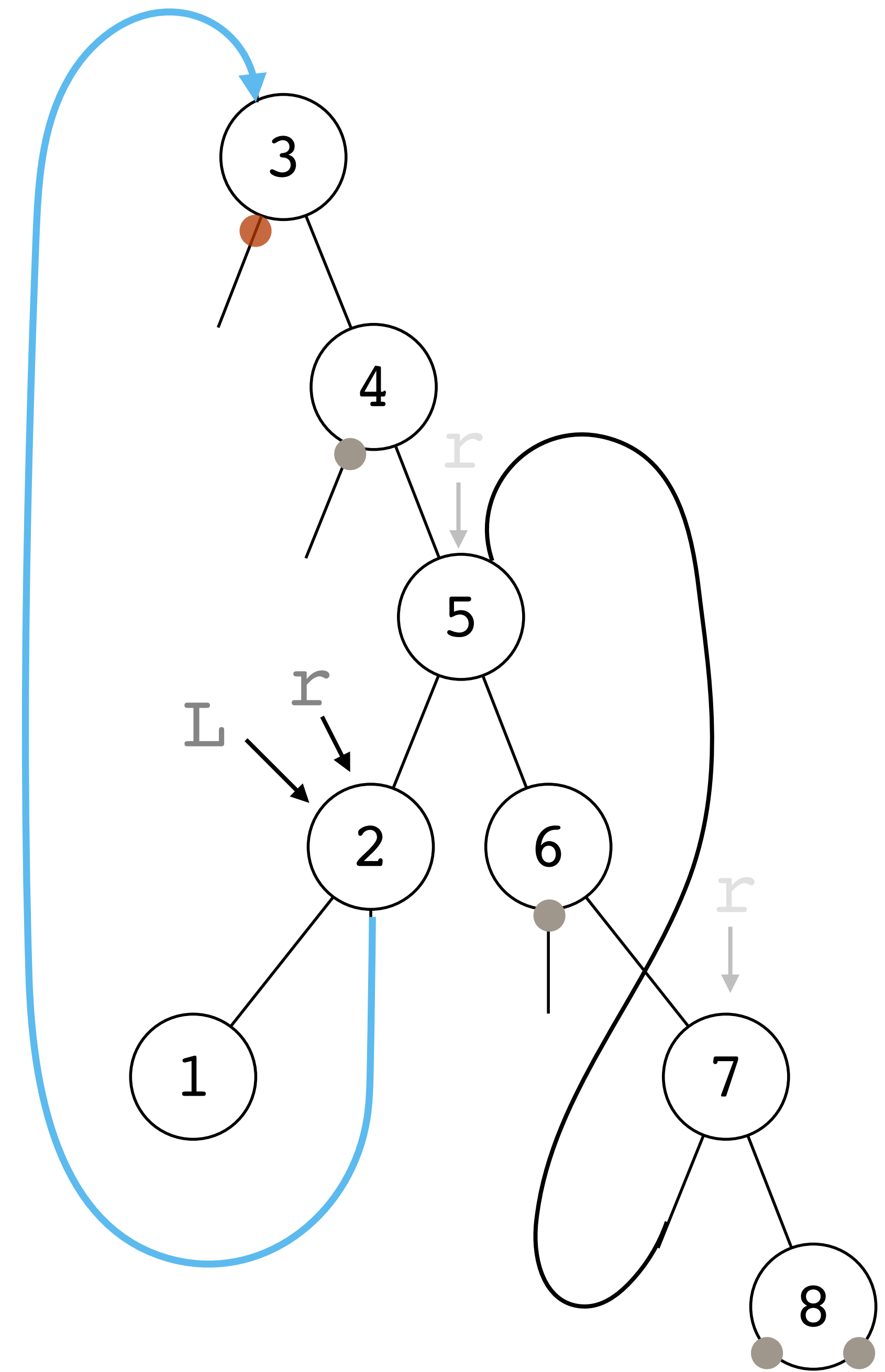
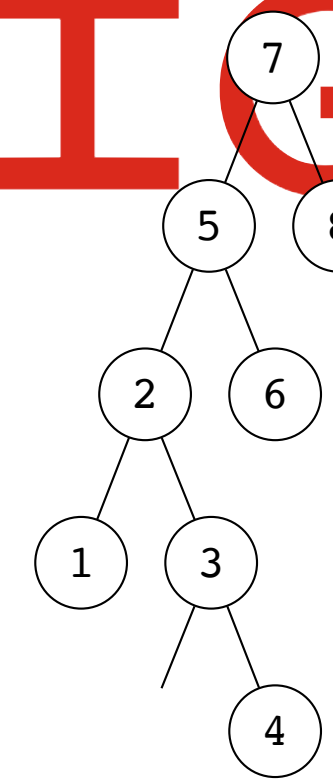
```

fonction linéariser (r, ref L, ref n)
    si r != ∅,
        linéariser(r.droit, L, n)
        r.droit ← L,    L ← r,    n ← n + 1
        linéariser(r.gauche, L, n)
    r.gauche ← ∅

```

HEIG

VD



5.droit $\leftarrow$ 6, L $\leftarrow$ 5, n $\leftarrow$ 4

linéariser(r=2, L=5, n=4)

linéariser(r=3, L=5, n=4)

linéariser(r=4, L=5, n=4)

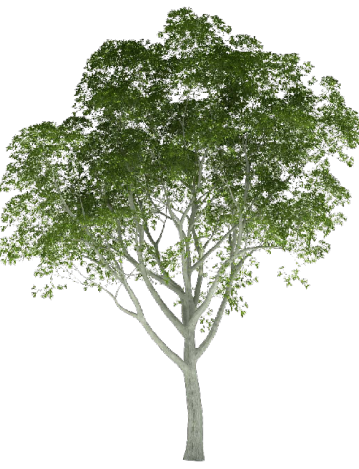
4.droit $\leftarrow$ 5, L $\leftarrow$ 4, n $\leftarrow$ 5

4.gauche $\leftarrow$ $\emptyset$

3.droit $\leftarrow$ 4, L $\leftarrow$ 3, n $\leftarrow$ 6

3.gauche $\leftarrow$ $\emptyset$

2.droit $\leftarrow$ 3, L $\leftarrow$ 2, n $\leftarrow$ 7



```

fonction linéariser (r, ref L, ref n)
  si r !=  $\emptyset$ ,
    linéariser(r.droit, L, n)
    r.droit  $\leftarrow$  L, L  $\leftarrow$  r, n  $\leftarrow$  n + 1
    linéariser(r.gauche, L, n)
    r.gauche  $\leftarrow$   $\emptyset$ 
  
```



1. **droit** $\leftarrow 2$, **L** $\leftarrow 1$, **n** $\leftarrow 7$

```

fonction linéariser (r, ref L, ref n)
    si r != ∅,
        linéariser(r.droit, L, n)
        r.droit ← L,    L ← r,    n ← n + 1
        linéariser(r.gauche, L, n)
        r.gauche ← ∅

```

linéariser(r=3, L=5, n=4)

linéariser(r=4, L=5, n=4)

```
4.droit ← 5,  L ← 4,  n ← 5
```

4.gauche $\leftarrow \emptyset$

3.droit $\leftarrow 4$, L $\leftarrow 3$, n $\leftarrow 6$

3.gauche $\leftarrow \emptyset$

$$2.\text{droit} \leftarrow 1, \quad L \leftarrow 2, \quad n \leftarrow 7$$

linéariser(r=1, L=2, n=7)

```
1. droit  $\leftarrow 2,$    L  $\leftarrow 1,$    n  $\leftarrow 7$ 
```



```

fonction linéariser (r, ref L, ref n)
    si r != ∅,
        linéariser(r.droit, L, n)
        r.droit ← L,    L ← r,    n ← n + 1
        linéariser(r.gauche, L, n)
        r.gauche ← ∅

```

linéariser(r=3, L=5, n=4)

linéariser(r=4, L=5, n=4)

```
4.droit ← 5,  L ← 4,  n ← 5
```

4.gauche $\leftarrow \emptyset$

3.droit $\leftarrow 4$, L $\leftarrow 3$, n $\leftarrow 6$

3.gauche $\leftarrow \emptyset$

```
2.droit ← 1,  L ← 2,  n ← 7
```

linéariser(r=1, L=2, n=7)

```
1. droit  $\leftarrow 2,$  L  $\leftarrow 1,$  n  $\leftarrow 7$ 
```

```
fonction linéariser (r, ref L, ref n)
```

si $r \neq \emptyset$,

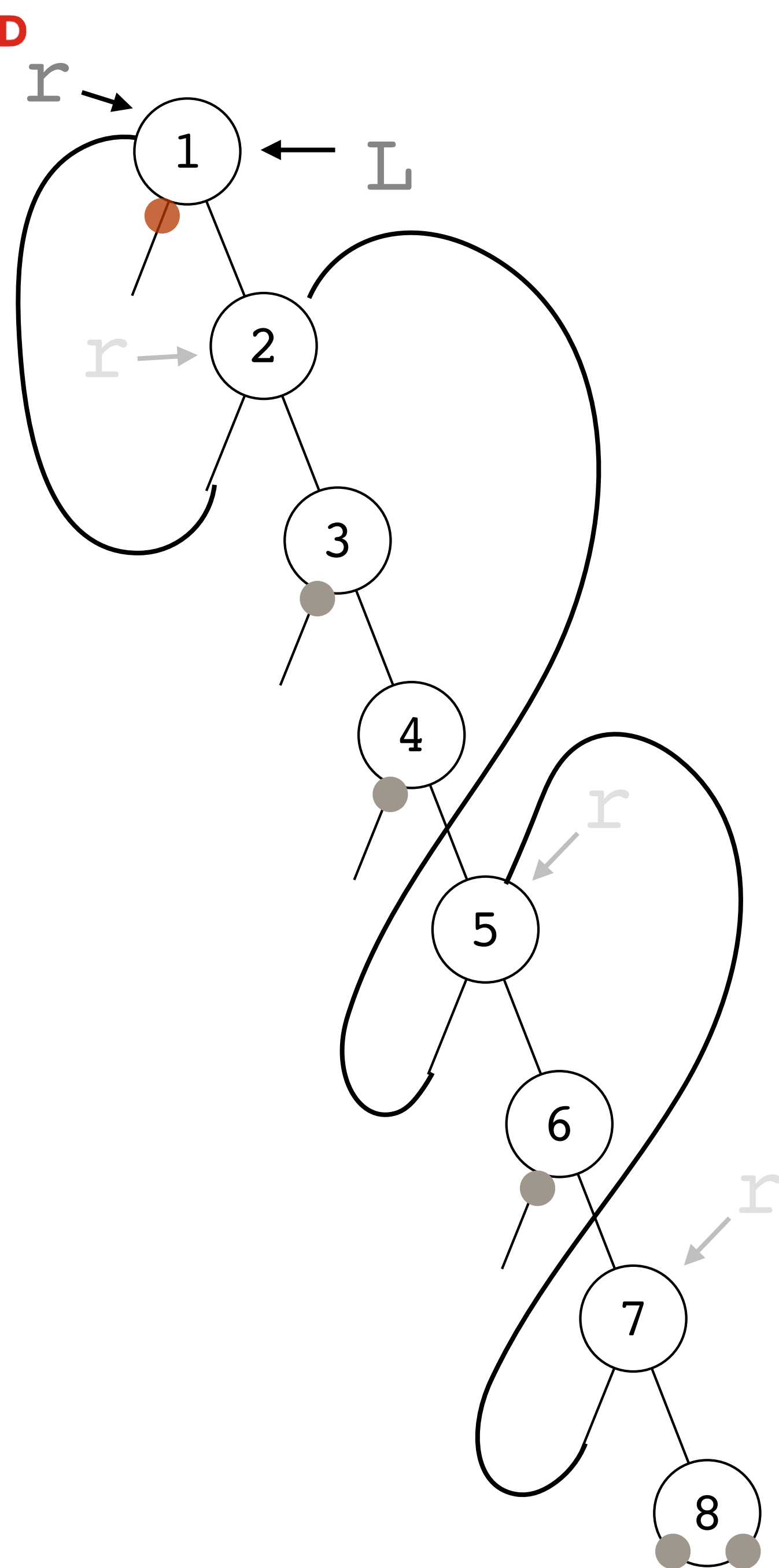
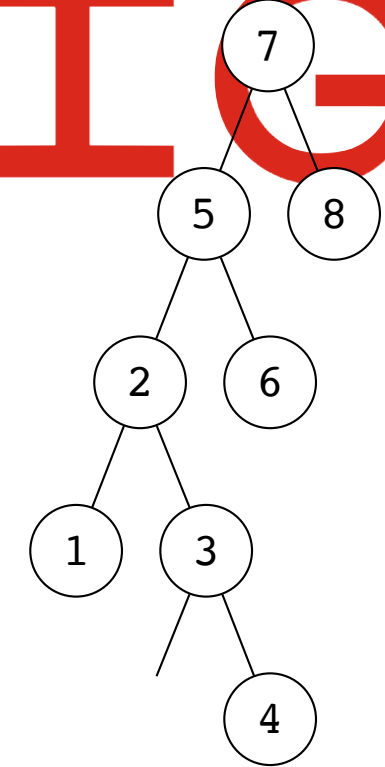
```
linéariser(r.droit, L, n)
```

```
r.droit ← L,   L ← r,   n ← n + 1
```

```
linéariser(r.gauche, L, n)
```

```
r.gauche ← ∅
```

HEIG



3.droit $\leftarrow$ 4, L $\leftarrow$ 3, n $\leftarrow$ 6

3.gauche $\leftarrow$ $\emptyset$

2.droit $\leftarrow$ 1, L $\leftarrow$ 2, n $\leftarrow$ 7

linéariser(r=1, L=2, n=7)

1.droit $\leftarrow$ 2, L $\leftarrow$ 1, n $\leftarrow$ 7

1.gauche $\leftarrow$ $\emptyset$

2.gauche $\leftarrow$ $\emptyset$

5.gauche $\leftarrow$ $\emptyset$

7.gauche $\leftarrow$ $\emptyset$

```
fonction linéariser (r, ref L, ref n)
```

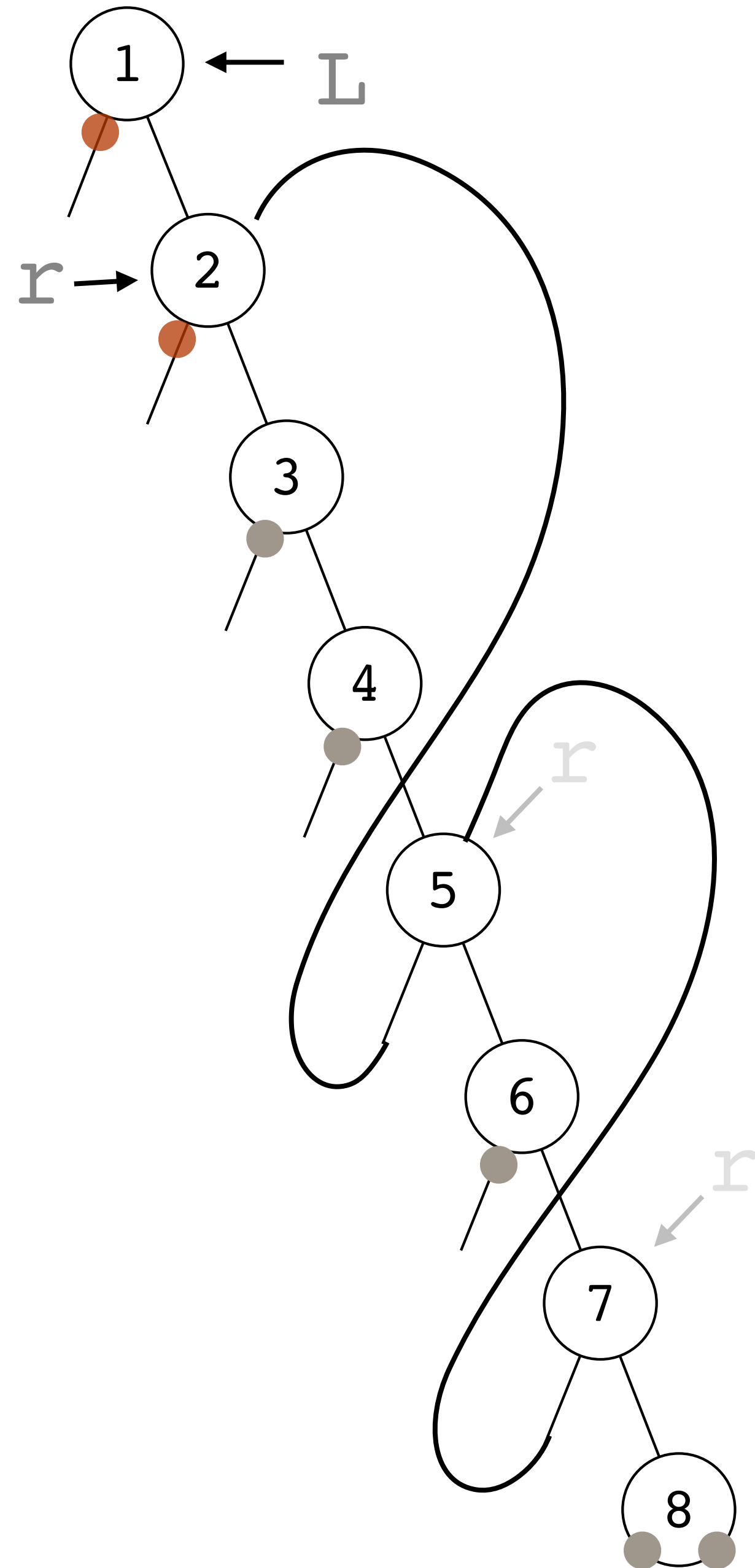
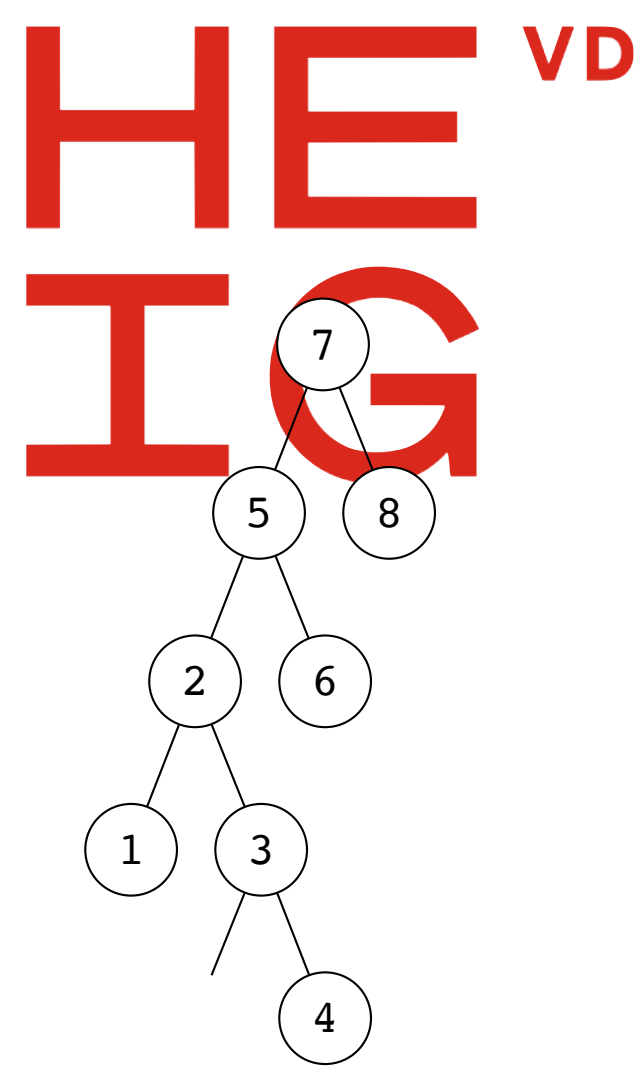
```
si r  $\neq$   $\emptyset$ ,
```

```
    linéariser(r.droit, L, n)
```

```
    r.droit  $\leftarrow$  L, L  $\leftarrow$  r, n  $\leftarrow$  n + 1
```

```
    linéariser(r.gauche, L, n)
```

```
    r.gauche  $\leftarrow$   $\emptyset$ 
```



3.droit $\leftarrow$ 4, L $\leftarrow$ 3, n $\leftarrow$ 6

3.gauche $\leftarrow$ $\emptyset$

2.droit $\leftarrow$ 1, L $\leftarrow$ 2, n $\leftarrow$ 7

linéariser(r=1, L=2, n=7)

1.droit $\leftarrow$ 2, L $\leftarrow$ 1, n $\leftarrow$ 7

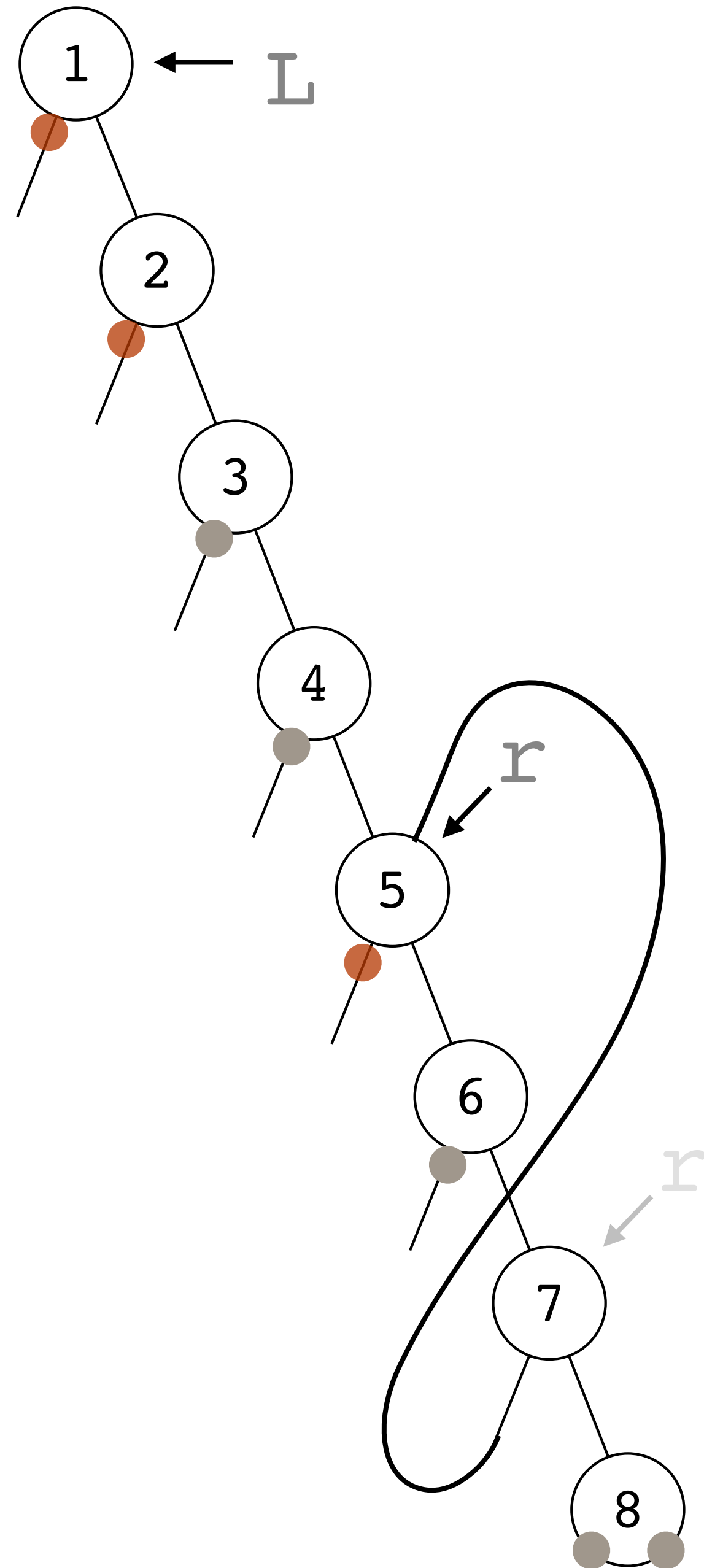
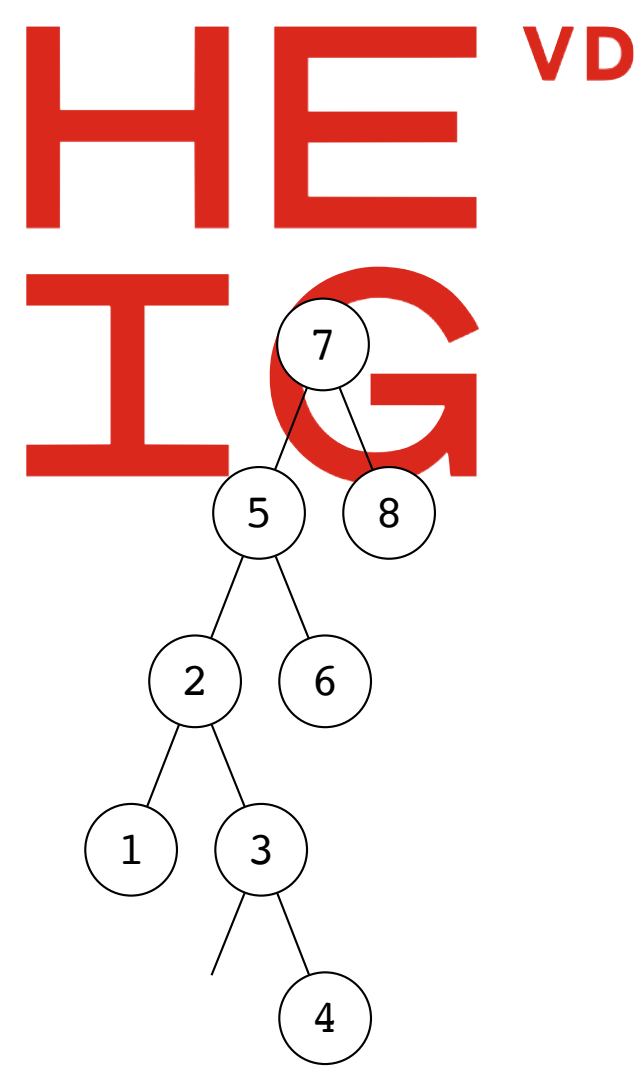
1.gauche $\leftarrow$ $\emptyset$

2.gauche $\leftarrow$ $\emptyset$

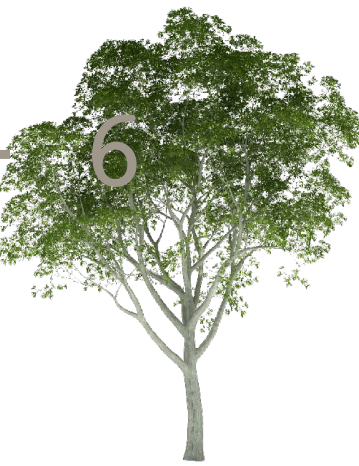
5.gauche $\leftarrow$ $\emptyset$

7.gauche $\leftarrow$ $\emptyset$

```
fonction linéariser (r, ref L, ref n)
  si r !=  $\emptyset$ ,
    linéariser(r.droit, L, n)
    r.droit  $\leftarrow$  L, L  $\leftarrow$  r, n  $\leftarrow$  n + 1
    linéariser(r.gauche, L, n)
    r.gauche  $\leftarrow$   $\emptyset$ 
```

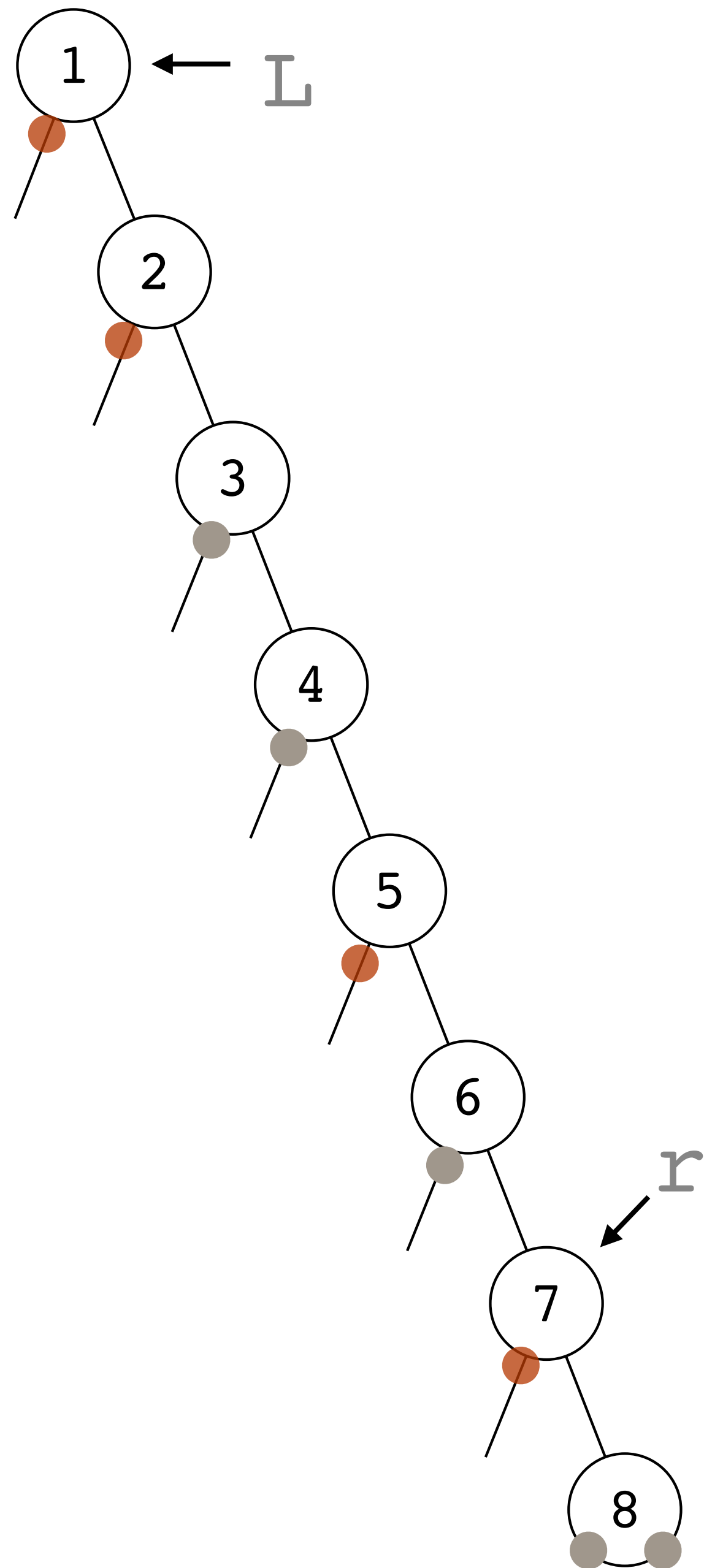
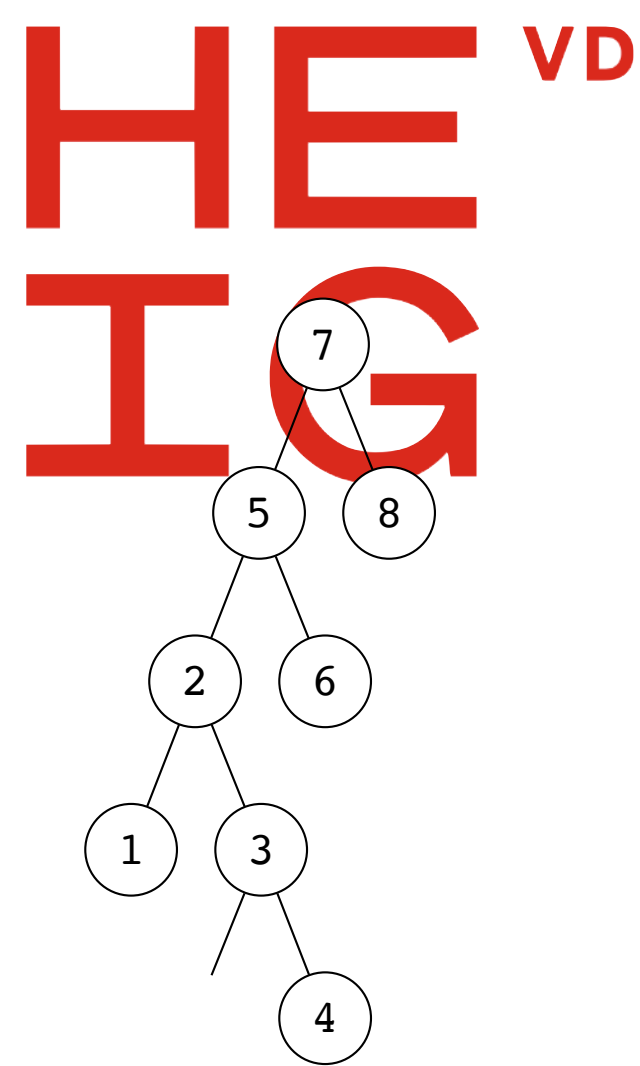


$3.\text{droit} \leftarrow 4, \quad L \leftarrow 3, \quad n \leftarrow 6$
 $3.\text{gauche} \leftarrow \emptyset$
 $2.\text{droit} \leftarrow 1, \quad L \leftarrow 2, \quad n \leftarrow 7$
 $\text{linéariser}(r=1, L=2, n=7)$
 $1.\text{droit} \leftarrow 2, \quad L \leftarrow 1, \quad n \leftarrow 7$
 $1.\text{gauche} \leftarrow \emptyset$
 $2.\text{gauche} \leftarrow \emptyset$
 $5.\text{gauche} \leftarrow \emptyset$
 $7.\text{gauche} \leftarrow \emptyset$

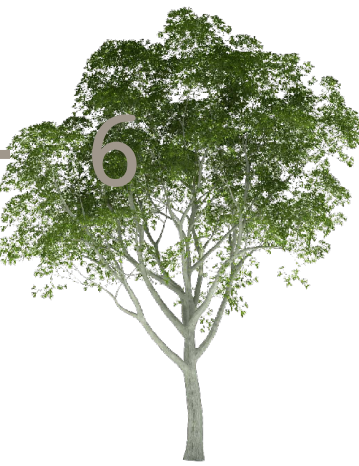


```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L,   L ← r,   n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
  
```



$3.\text{droit} \leftarrow 4, \quad L \leftarrow 3, \quad n \leftarrow 6$
 $3.\text{gauche} \leftarrow \emptyset$
 $2.\text{droit} \leftarrow 1, \quad L \leftarrow 2, \quad n \leftarrow 7$
 $\text{linéariser}(r=1, L=2, n=7)$
 $1.\text{droit} \leftarrow 2, \quad L \leftarrow 1, \quad n \leftarrow 7$
 $1.\text{gauche} \leftarrow \emptyset$
 $2.\text{gauche} \leftarrow \emptyset$
 $5.\text{gauche} \leftarrow \emptyset$
 $7.\text{gauche} \leftarrow \emptyset$



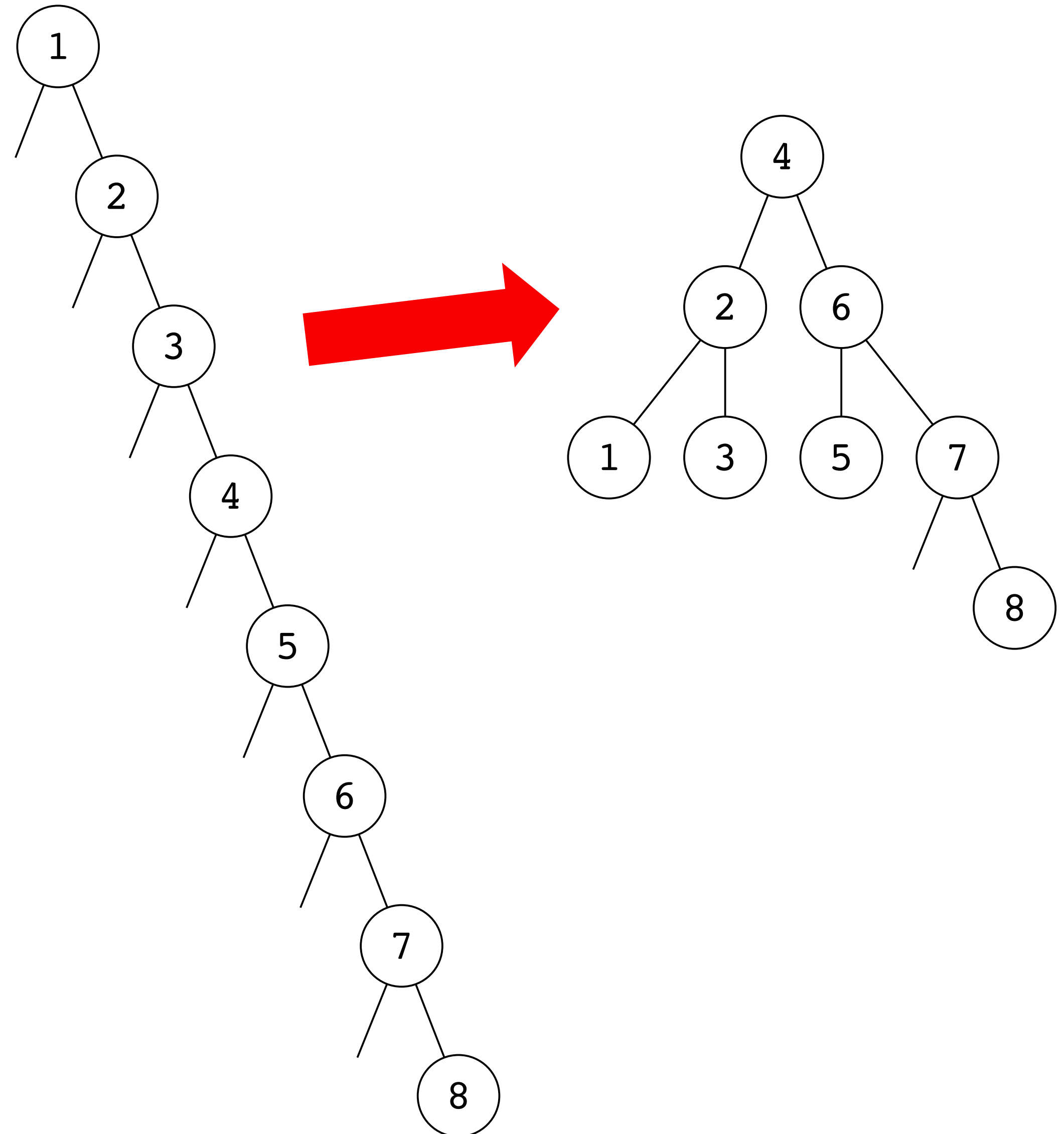
```

fonction linéariser (r, ref L, ref n)
  si r != ∅,
    linéariser(r.droit, L, n)
    r.droit ← L,   L ← r,   n ← n + 1
    linéariser(r.gauche, L, n)
    r.gauche ← ∅
  
```

Arborisation



Réorganisation d'un arbre dégénéré
à droite en un arbre plein.



Arborisation



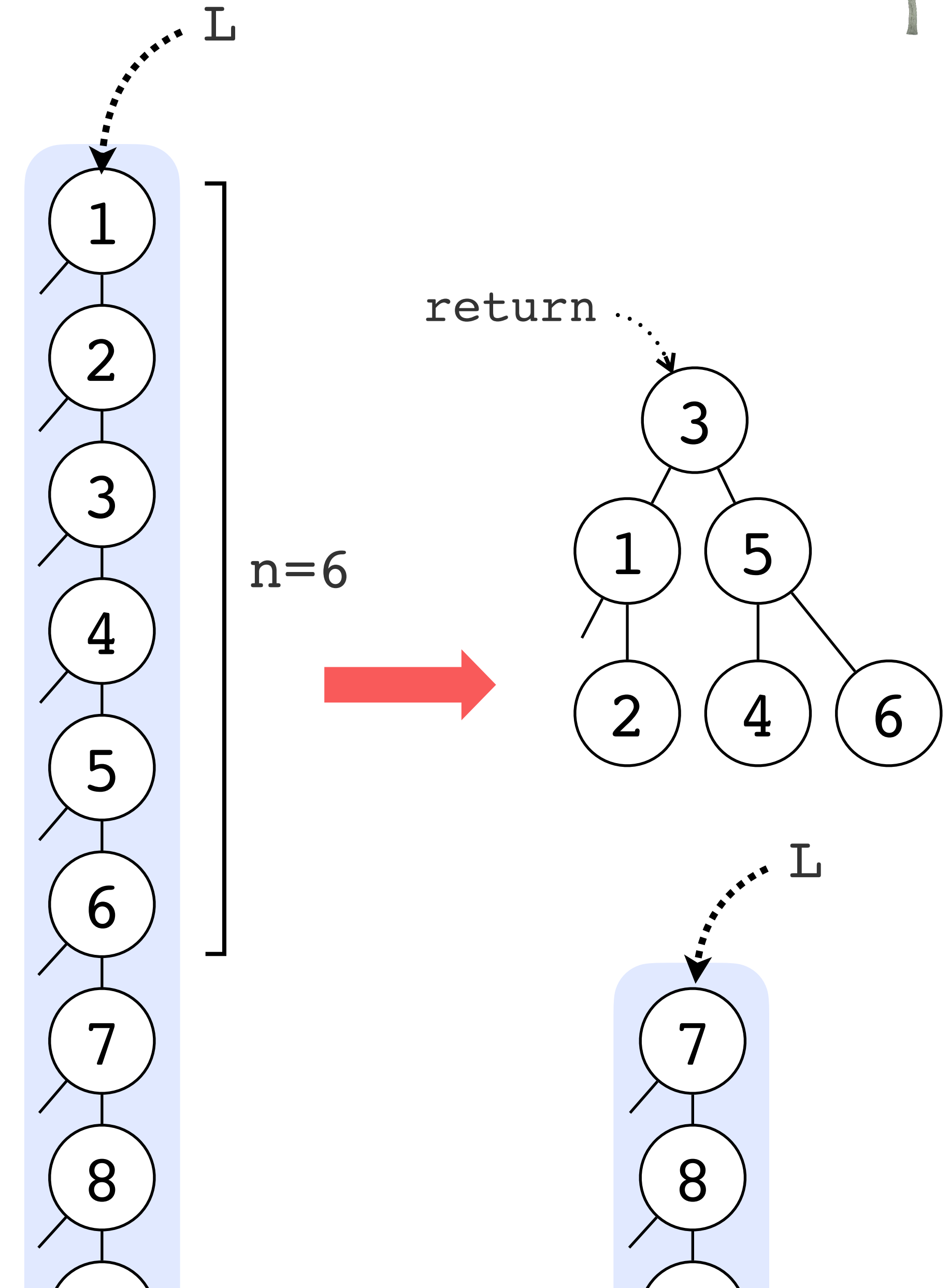
Paramètres :

(*inout*) Lien $\mathbb{L}$ vers l'arbre sans enfant gauche.

(*in*) Nombre n d'éléments à traiter.

Modifie $\mathbb{L}$ pour pointer sur le $n+1^{\text{ième}}$ noeud.

Retourne la racine de l'arborisation
des n premiers noeuds.



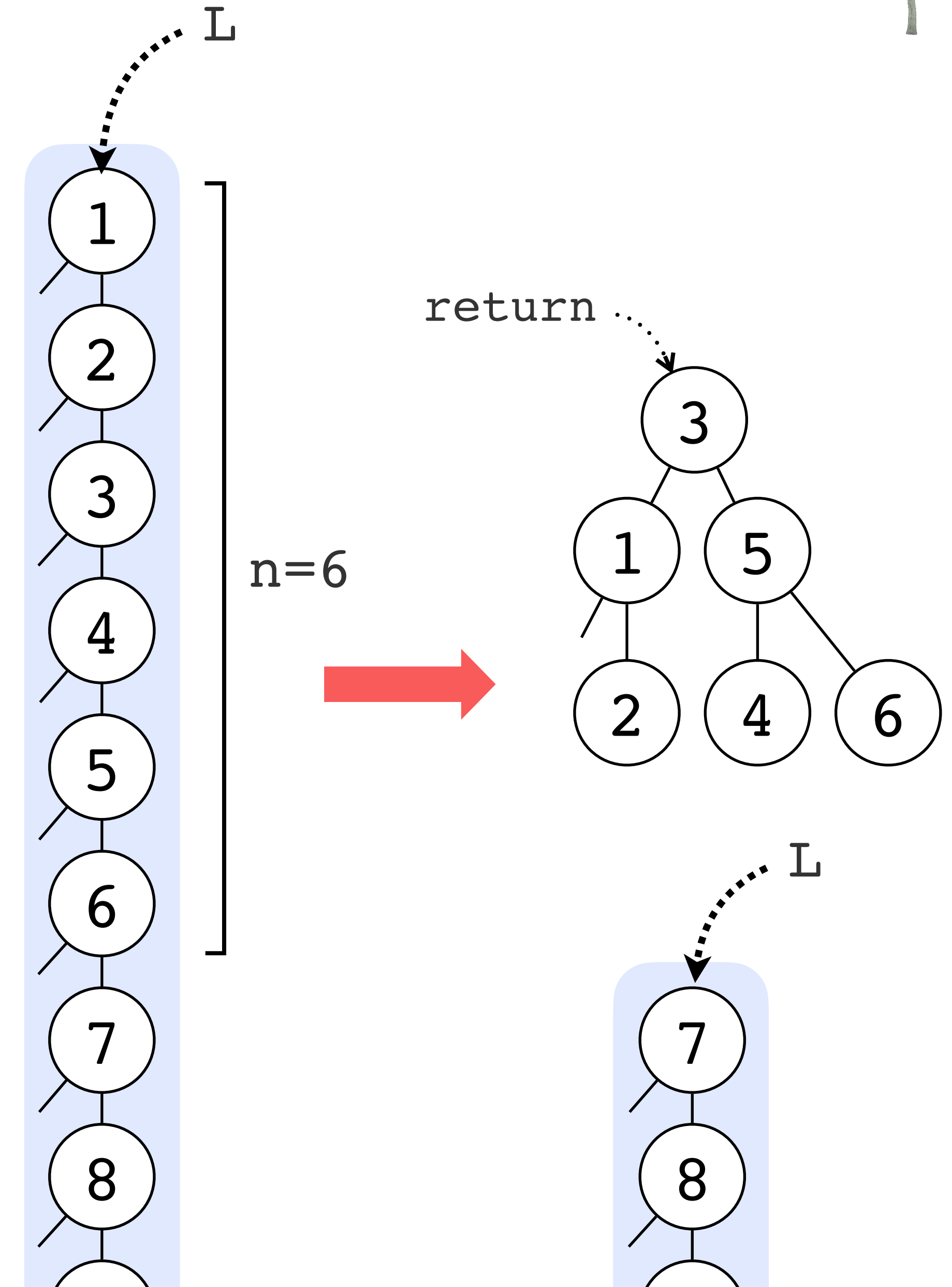
Arborisation



Idée

- Prendre l'élément médian comme racine
- Arboriser récursivement les sous-arbres gauche et droit

Comment trouver la médiane ?



Arborisation



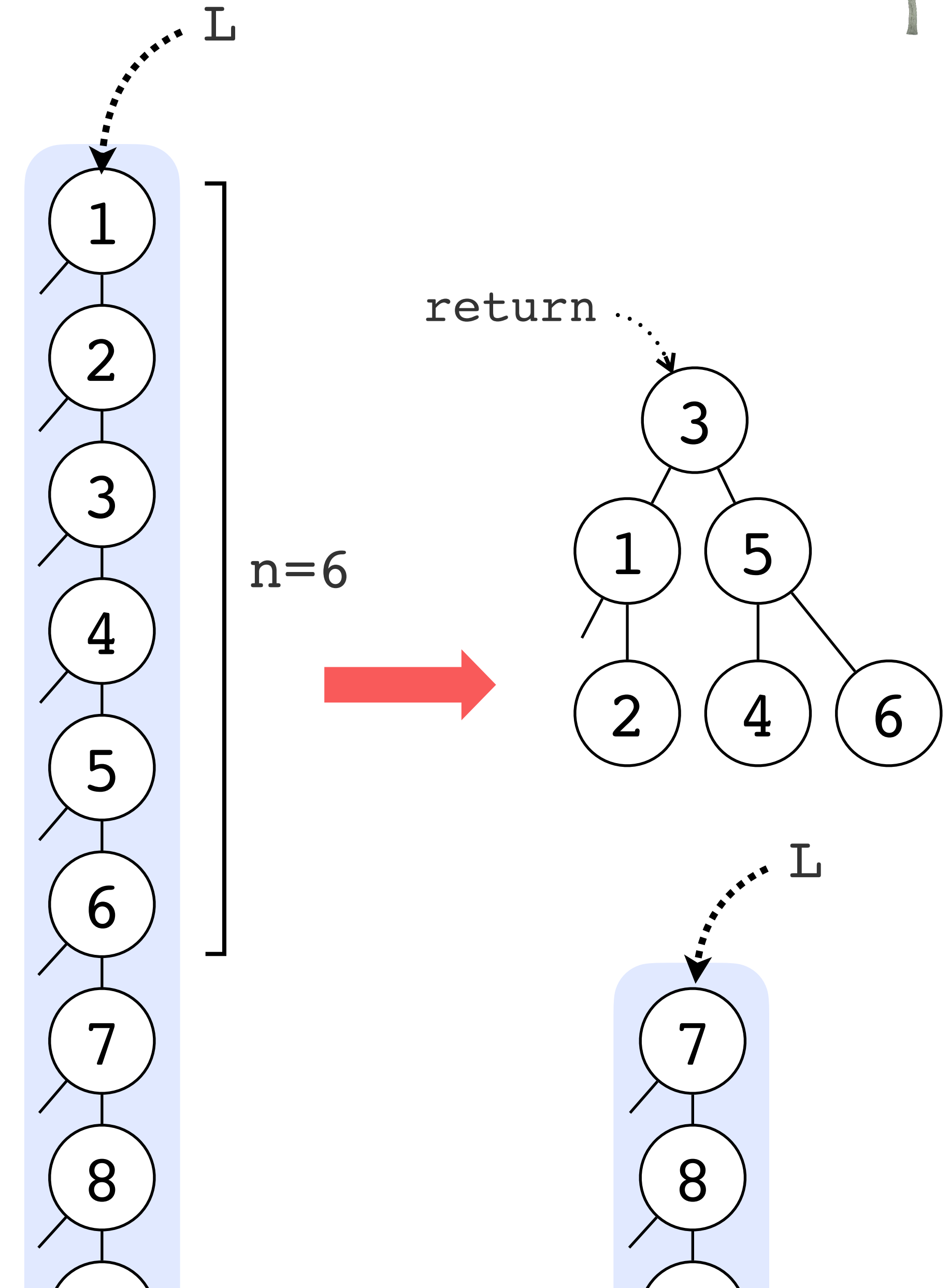
Idée

- Prendre l'élément médian comme racine
- Arboriser récursivement les sous-arbres gauche et droit

Comment trouver la médiane ?

- Faire l'appel récursif sur les $(n-1)/2$ premiers éléments (futur sous-arbre gauche)
- $\mathbb{L}$ pointe alors sur la médiane

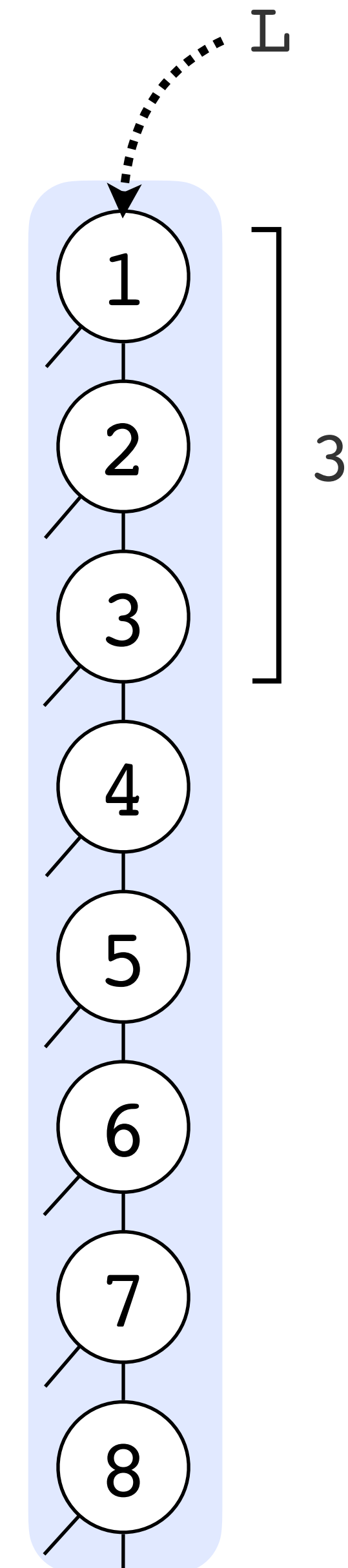
Revient à parcours symétrique de l'arbre final.



Arborisation



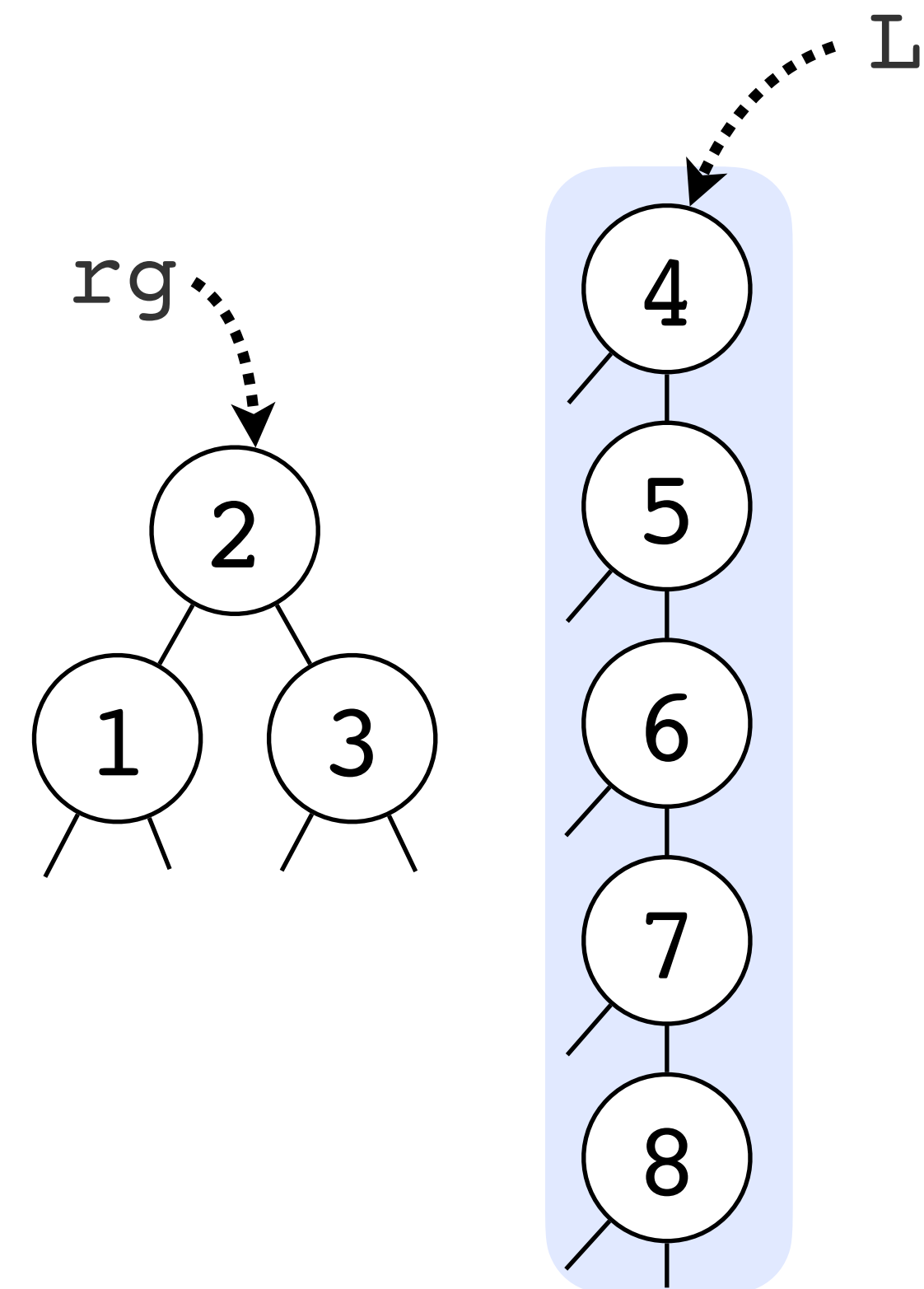
```
fonction arboriser (ref L, n)
  si n != 0,
    rg ← arboriser(L, (n-1)/2) (div. entière)
    r ← L, r.gauche ← rg, L ← L.droit
    r.droit ← arboriser(L, n/2) (div. entière)
  retourner r
```



Arborisation



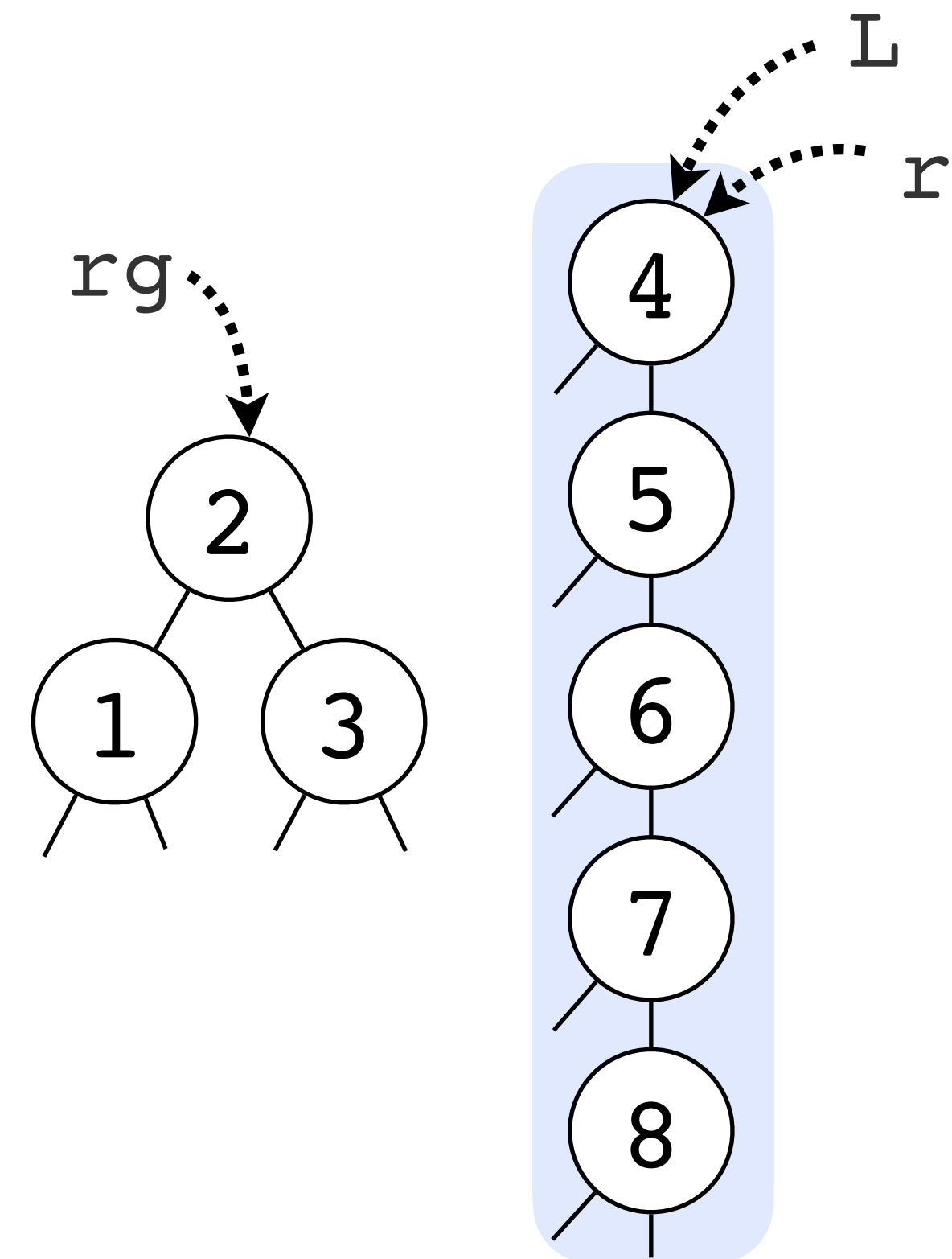
```
fonction arboriser (ref L, n)
  si n != 0,
    rg ← arboriser(L, (n-1)/2) (div. entière)
    r ← L, r.gauche ← rg, L ← L.droit
    r.droit ← arboriser(L, n/2) (div. entière)
  retourner r
```



Arborisation



```
fonction arboriser (ref L, n)
  si n != 0,
    rg ← arboriser(L, (n-1)/2) (div. entière)
    r ← L, r.gauche ← rg, L ← L.droit
    r.droit ← arboriser(L, n/2) (div. entière)
  retourner r
```

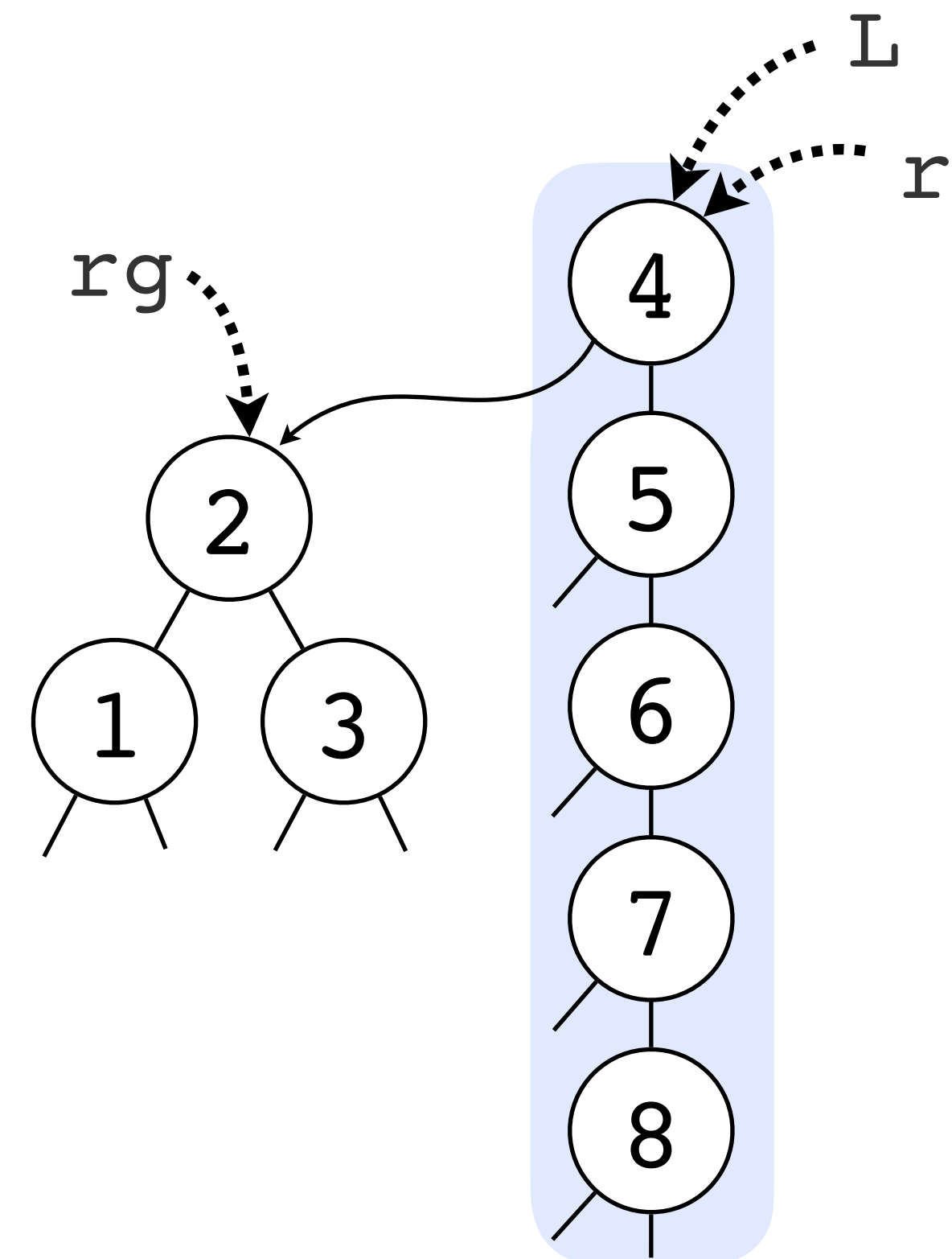


Arborisation



```

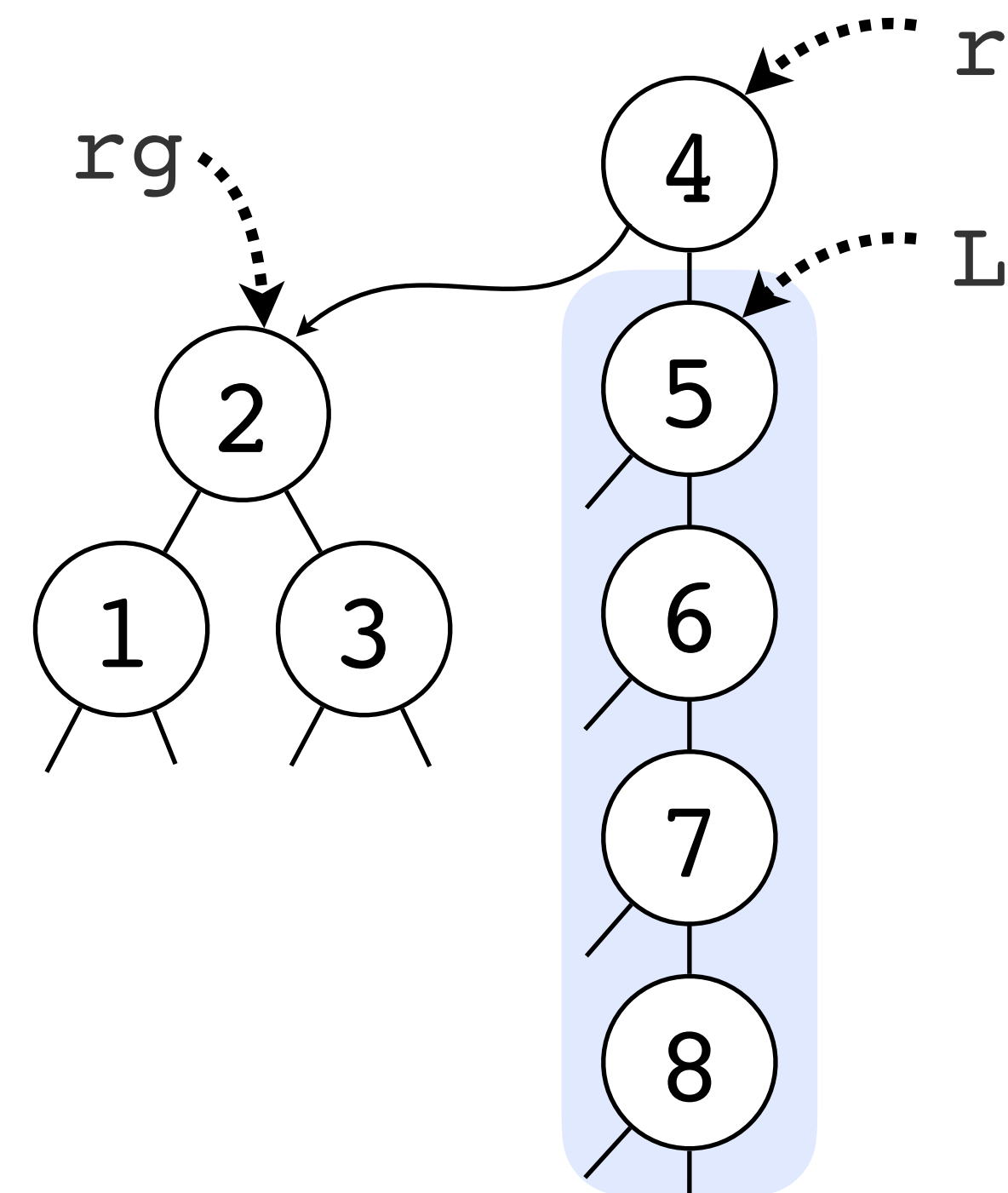
fonction arboriser (ref L, n)
  si n != 0,
    rg ← arboriser(L, (n-1)/2) (div. entière)
    r ← L, r.gauche ← rg, L ← L.droit
    r.droit ← arboriser(L, n/2) (div. entière)
  retourner r
  
```



Arborisation



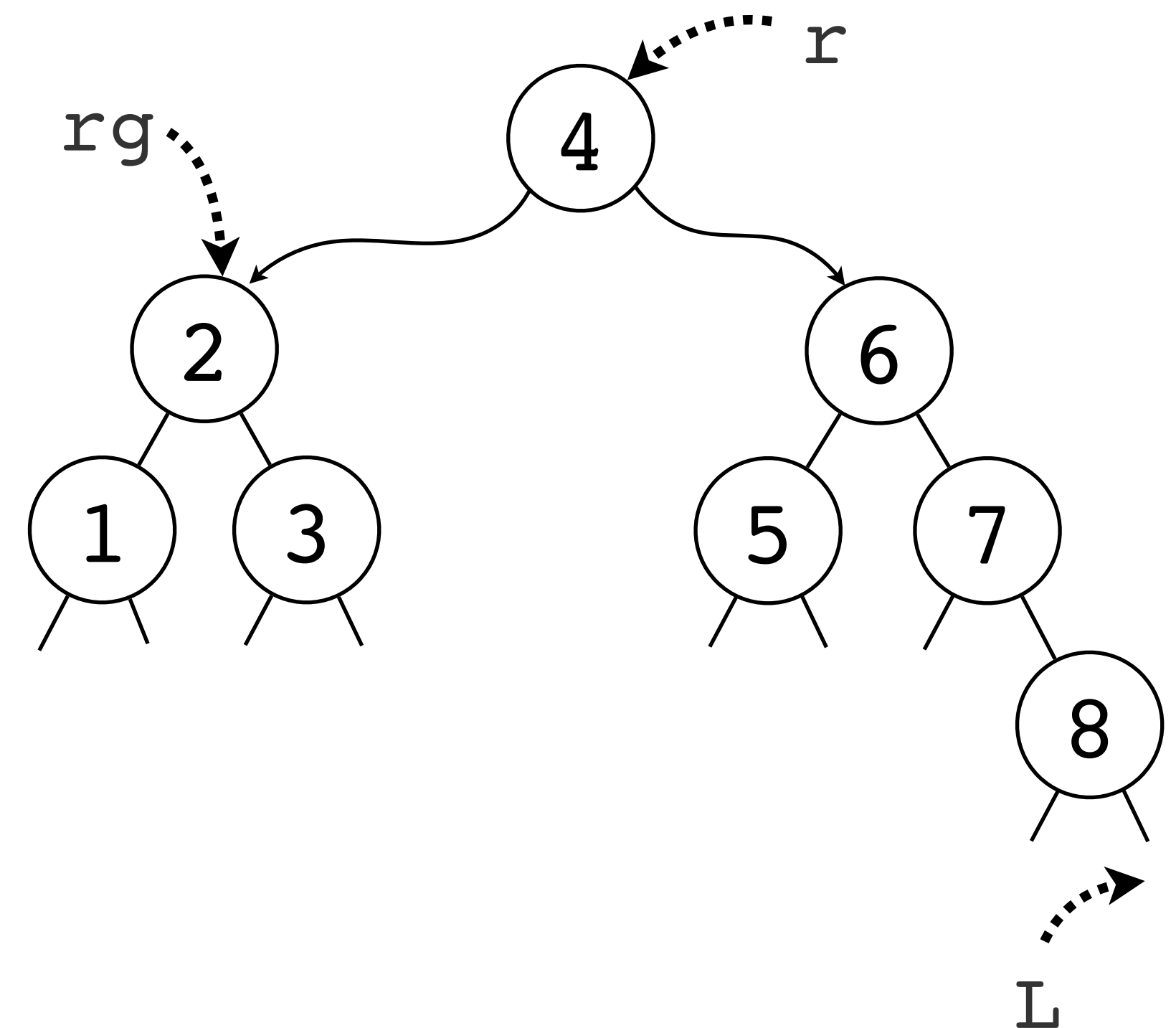
```
fonction arboriser (ref L, n)
  si n != 0,
    rg ← arboriser(L, (n-1)/2) (div. entière)
    r ← L, r.gauche ← rg, L ← L.droit
    r.droit ← arboriser(L, n/2) (div. entière)
  retourner r
```



Arborisation



```
fonction arboriser (ref L, n)
  si n != 0,
    rg ← arboriser(L, (n-1)/2) (div. entière)
    r ← L, r.gauche ← rg, L ← L.droit
    r.droit ← arboriser(L, n/2) (div. entière)
  retourner r
```





- Linéarisation puis arborisation

```
fonction équilibrer (r)
  L ← ∅,  n ← 0
  linéariser(r, L, n)
  retourner arboriser(L, n)
```

- Les 2 algorithmes sont des parcours de l'arbre, donc de complexité $O(n)$