

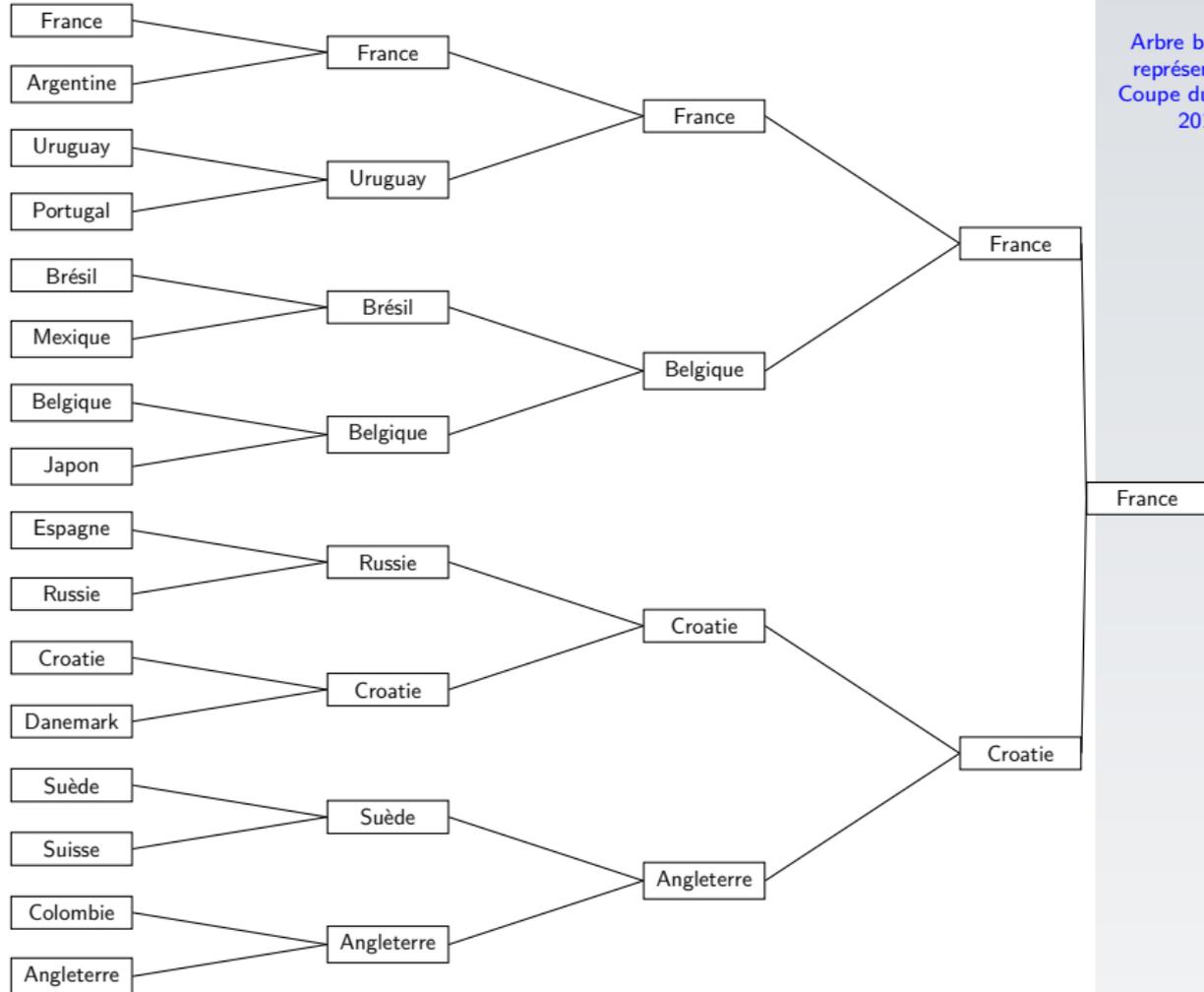
# Arbre binaire - représentation Coupe du monde 2018

Christophe Viroulaud

Terminale - NSI

**Algo 07**

La France a (brillamment) gagné la coupe du Monde de football 2018. Pour stocker le tableau des phases finales le site web *lateam.fr* utilise une structure simple de données.



Comment représenter simplement un arbre binaire en  
mémoire ?

L'arbre binaire (figure 1) est numéroté en utilisant un parcours en largeur.

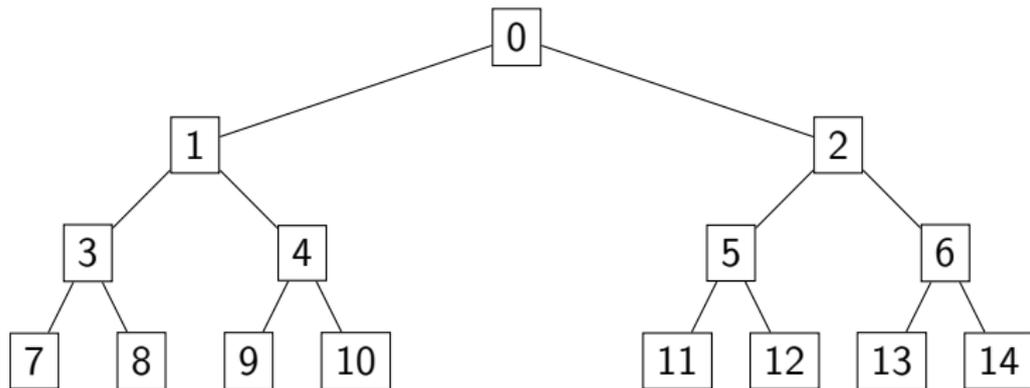
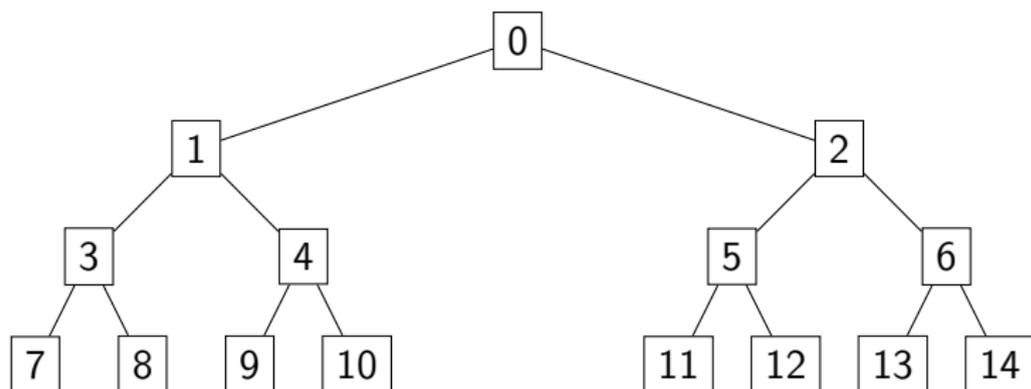


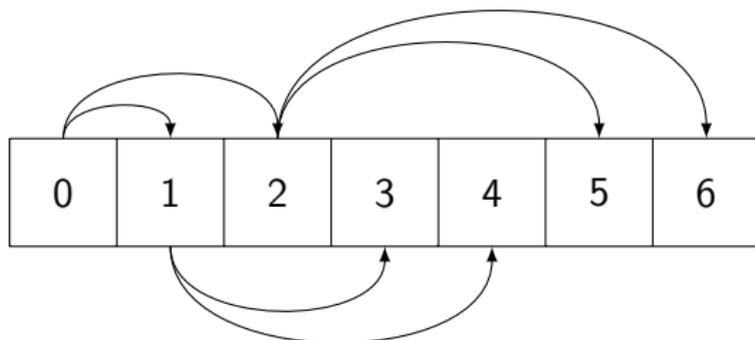
FIGURE 1 – Relation entre les indices



Pour chaque nœud  $i$  qui a des fils, on peut remarquer que :

- ▶ l'indice du fils gauche est  $2 \times i + 1$ ,
- ▶ l'indice du fils droit est  $2 \times i + 2$ .

Un arbre binaire peut alors être stocké dans un simple tableau.



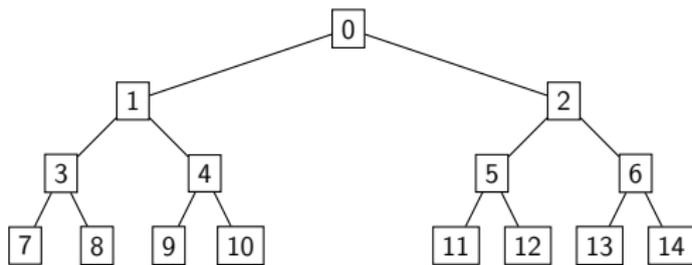
Code 1 – Un arbre binaire dans un tableau

## Activité 1 :

1. Télécharger le fichier *cdm2018.zip* sur le site <https://cviroulaud.github.io> .
2. Ouvrir le fichier *cdm2018.json* et vérifier à la main que le tableau représente bien l'arbre binaire des phases finales de la coupe du Monde 2018.
3. Importer le *json* dans un programme Python.

```
1 f = open("cdm2018.json")
2 tab_cdm = json.load(f)
3 f.close()
```

Code 2 – Import



**Activité 2 :** Considérons un arbre binaire parfait de hauteur  $h$  représenté par un tableau.

1. Quel est sa taille ?
2. Combien y-a-t-il de feuilles ?
3. Quel est l'indice de la feuille la plus à gauche ?
4. Écrire la fonction `i_feuille_gauche(arbre: list) → int` qui renvoie l'indice de la feuille la plus à gauche.
5. Écrire alors la fonction `get_matches(arbre: list) → list` qui renvoie la liste des matchs de huitième de finale sous la forme d'un tableau de tuples.

- ▶ La taille est  $N = 2^{(h+1)} - 1$ .
- ▶ Il y a  $2^h$  feuilles.
- ▶ L'indice de la première feuille est  $2^h - 1$ .

```
1 def i_feuille_gauche(arbre: list) -> int:
2     """
3     indice de la feuille la plus à gauche
4
5     Args:
6         arbre (list): arbre binaire parfait
7
8     Returns:
9         int: l'indice
10    """
11    i = 0
12    while (2*i+1) < len(arbre):
13        i = 2*i+1
14    return i
```

```
1 def get_matches(arbre: list) -> list:
2     """
3     huitième de finale
4
5     Args:
6         arbre (list): tableau du tournoi
7
8     Returns:
9         list: tableau de tuples
10    """
11    matchs = []
12    i = i_feuille_gauche(arbre)
13    while i < len(arbre):
14        matchs.append((arbre[i], arbre[i+1]))
15        i = i+2
16    return matchs
```