Structures de données : les arbres:

Un organisateur de tournoi de rugby recherche la meilleure solution pour afficher les potentiels quarts de final, demi-finales et finale:

Au départ nous avons 4 poules de 4 équipes. Les 4 équipes d'une poule s'affrontent dans un mini championnat (3 matchs par équipe). À l'issue de cette phase de poule, les 2 premières équipes de chaque poule sont qualifiées pour les quarts de finale. Dans ce qui suit, on désigne les 2 qualifiés par poule par:

- Poule 1 $\longrightarrow$ 1er Eq1 ; 2e Eq8
- Poule 2 $\longrightarrow$ 1er Eq2 ; 2e Eq7
- Poule 3 $\longrightarrow$ 1er Eq3 ; 2e Eq6
- Poule 4 $\longrightarrow$ 1er Eq4 ; 2e Eq5

En quart de final on va avoir:

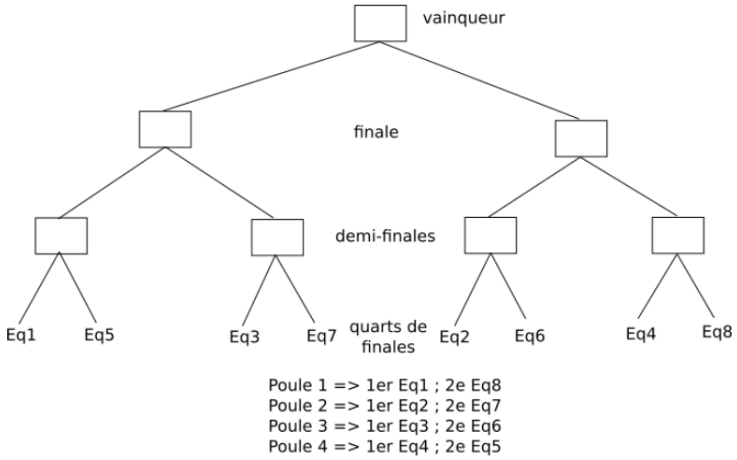
- quart de finale 1 $\longrightarrow$ Eq1 contre Eq5
- quart de finale 2 $\longrightarrow$ Eq2 contre Eq6
- quart de finale 3 $\longrightarrow$ Eq3 contre Eq7
- quart de finale 4 $\longrightarrow$ Eq4 contre Eq8

Pour les demi-finales on aura:

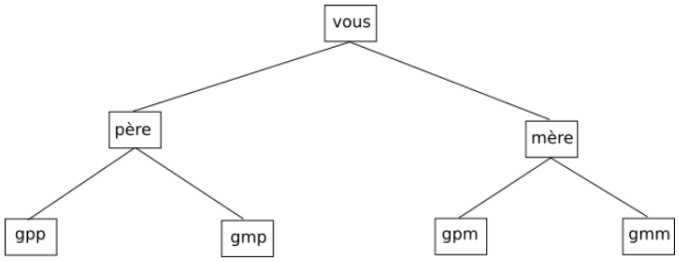
- demi-final 1 $\longrightarrow$ vainqueur quart de finale 1 contre vainqueur quart de finale 3
- demi-final 2 $\longrightarrow$ vainqueur quart de finale 2 contre vainqueur quart de finale 4

L'organisateur du tournoi affiche les informations ci-dessus le jour du tournoi. Malheureusement, la plupart des spectateurs se perdent quand ils cherchent à déterminer les potentielles demi-finales (et ne parlons pas de la finale !)

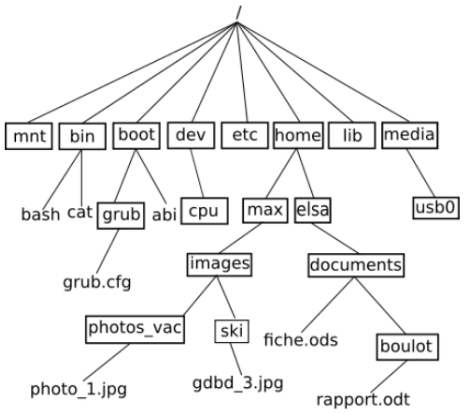
Pourtant, un simple graphique aurait grandement simplifié les choses:



Les spectateurs peuvent alors recopier sur un bout de papier ce schéma et ensuite se livrer au jeu des pronostiques. Nous avons ci-dessous ce que l'on appelle une structure en arbre. On peut aussi retrouver cette même structure dans un arbre généalogique:



Dernier exemple, les systèmes de fichiers dans les systèmes de type UNIX ont aussi une structure en arbre (notion abordée en première: `https://pixees.fr/informatiquelycee/n_site/nsi_prem_cmd_base_linux.html`)

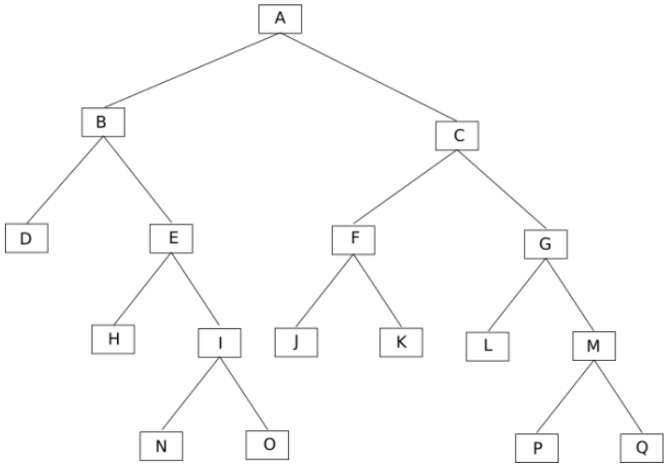


système de fichiers de type UNIX

Les arbres sont des types abstraits très utilisés en informatique. On les utilise notamment quand on a besoin d'une structure hiérarchique des données: dans l'exemple ci-dessous le fichier grub.cfg ne se trouve pas au même niveau que le fichier rapport.odt (le fichier grub.cfg se trouve "plus proche" de la racine / que le fichier rapport.odt). On ne pourrait pas avec une simple liste qui contiendrait les noms des fichiers et des répertoires, rendre compte de cette hiérarchie (plus ou moins "proche" de la racine). On trouve souvent dans cette hiérarchie une notion de temps (les quarts de finale sont avant les demi-finales ou encore votre grand-mère paternelle est née avant votre père), mais ce n'est pas une obligation (voir l'arborescence du système de fichiers).

Les arbres binaires sont des cas particuliers d'arbre : l'arbre du tournoi de rugby et l'arbre "père, mère..." sont des arbres binaires, en revanche, l'arbre représentant la structure du système de fichier n'est pas un arbre binaire. Dans un arbre binaire, on a au maximum 2 branches qui partent d'un élément (pour le système de fichiers, on a 7 branches qui partent de la racine : ce n'est donc pas un arbre binaire). Dans la suite nous allons uniquement travailler sur les arbres binaires.

Soit l'arbre binaire suivant:



Un peu de vocabulaire:

- chaque élément de l'arbre est appelé noeud (par exemple : A, B, C, D,...,P et Q sont des noeuds)
- le noeud initial (A) est appelé noeud racine ou plus simplement racine
- on dira que le noeud E et le noeud D sont les fils du noeud B. On dira que le noeud B est le père des noeuds E et D
- dans un arbre binaire, un noeud possède au plus 2 fils
- un noeud n'ayant aucun fils est appelé feuille (exemples : D, H, N, O, J, K, L, P et Q sont des feuilles)
- à partir d'un noeud (qui n'est pas une feuille), on peut définir un sous-arbre gauche et un sous-arbre droite (exemple : à partir de C on va trouver un sous-arbre gauche composé des noeuds F, J et K et un sous-arbre droit composé des noeuds G, L, M, P et Q)
- on appelle arête le segment qui relie 2 noeuds.

- on appelle profondeur d'un noeud ou d'une feuille dans un arbre binaire le nombre de noeuds du chemin qui va de la racine à ce noeud. La racine d'un arbre est à une profondeur 1, et la profondeur d'un noeud est égale à la profondeur de son prédécesseur plus 1. Si un noeud est à une profondeur p, tous ses successeurs sont à une profondeur p+1.
  Exemples: profondeur de B = 2 ; profondeur de I = 4 ; profondeur de P = 5

- on appelle hauteur d'un arbre la profondeur maximale des noeuds de l'arbre. Exemple : la profondeur de P = 5, c'est un des noeuds les plus profond, donc la hauteur de l'arbre est de 5.

Il est aussi important de bien noter que l'**on peut aussi voir les arbres comme des structures récursives**: les fils d'un noeud sont des arbres (sous-arbre gauche et un sous-arbre droite dans le cas d'un arbre binaire), ces arbres sont eux mêmes constitués d'arbres...

Exercice: Trouvez un autre exemple de données qui peuvent être représentées par un arbre binaire (dans le domaine de votre choix). Dessinez au moins une partie de cet arbre binaire. Déterminez la hauteur de l'arbre que vous aurez dessiné.

Python ne propose pas de façon native l'implémentation des arbres binaires. Mais nous aurons, plus tard dans l'année, l'occasion d'implémenter des arbres binaires en Python en utilisant un peu de programmation orientée objet. Nous aurons aussi très prochainement l'occasion d'étudier des algorithmes permettant de travailler sur les arbres binaires.