INFO-F103 – Algorithmique I
Projet 2 - La startup EOL

Youcef Bouharaoua
youcef.bouharaoua@ulb.be

Année académique 2021-2022

Binary EOL

Face aux défis environnementaux auxquels fait face notre planète et face à l'augmentation des prix des énergies fossiles, vous avez la bonne idée d'utiliser vos connaissances en informatique ainsi que votre esprit entrepreneurial afin de construire un parc éolien qui fonctionnera d'une manière innovante.

Ce parc aura la forme d'un arbre binaire où chaque noeud représente une éolienne. Chaque éolienne est identifiée par un nombre n qui lui est propre.

Fonctionnement

Vous décidez de créer un parc éolien intelligent. Pour activer toutes les éoliennes du parc, il vous suffit d'envoyer un unique signal à une éolienne, qui elle-même va envoyer le signal aux éoliennes directement connectées à celle-ci (parent, fils gauche et fils droit). Et ainsi de suite jusqu'à l'activation de tout le parc. Aussi, trois contraintes doivent être prises en compte:

- La première éolienne activée doit être une feuille.
- On ne peut activer qu'un seul noeud à la fois.
- Lorsque le signal est donné à un ensemble d'éoliennes, cet ensemble doit être complètement actif avant de lancer de nouveaux signaux.

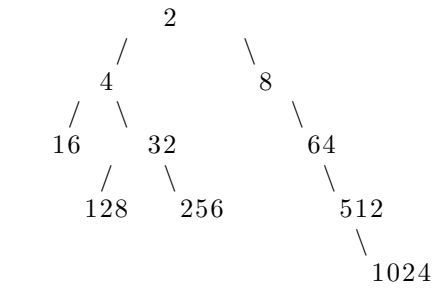
Pour mieux comprendre, regardons l'exemple suivant ensemble:

Considérons le noeud 256 de cet arbre, qui représente une éolienne du parc. En lui envoyant le signal de s'allumer celle-ci va prendre 256 secondes avant de s'allumer et de transmettre à son tour le signal d'allumage aux éoliennes directement reliées à elle (ici le noeud 32). Il faudra alors 32 secondes pour que l'éolienne 32 s'allume. A son tour, le noeud 32 va transmettre ce même signal aux noeuds directement liés à ce dernier c'est à dire, le 4 et le 128. Ces derniers s'allumeront respectivement après 4 et 128 secondes. Et ainsi de suite.

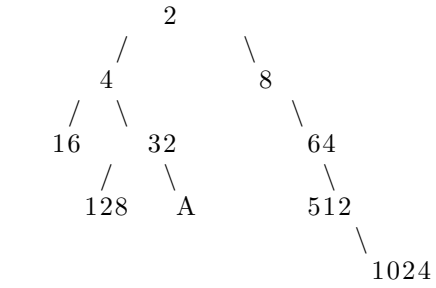
Ainsi, à T= 256s, seule l'éolienne 256 est allumée et il faudra 2046 secondes pour allumer tout le parc.

Votre mission consiste donc à trouver le temps nécessaire pour allumer tout le parc d'éoliennes (l'arbre binaire complet).

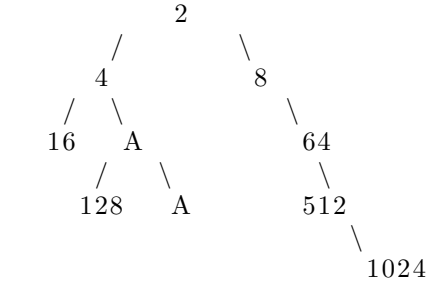
Input :



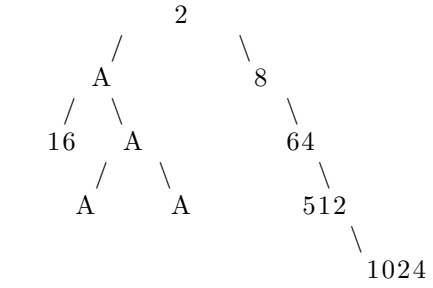
Feuille initiale (cible) = 256
Output : 2046 secondes
L'eolienne 256 se met en marche apres 256s.



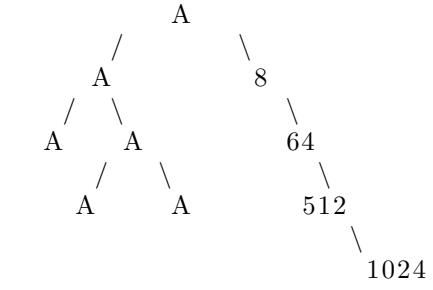
T= 288s : l'eolienne 32 s'active



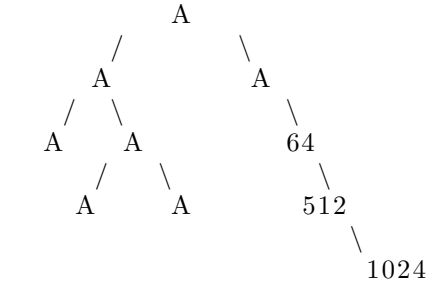
T = 292s et T = 420s (respec.) : les eoliennes 4 et 128 s'activent.



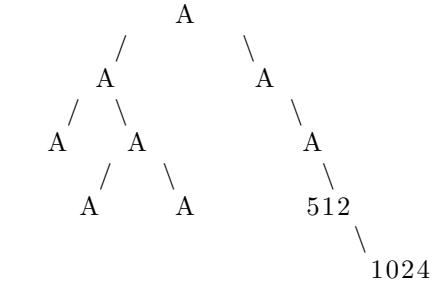
T = 436s et T= 438s (respec.): les eoliennes 16 et 2 s'activent.



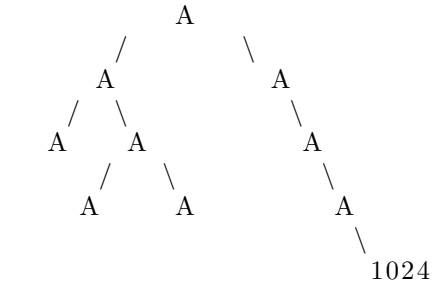
T = 446s : l'eolienne 8 s'active.



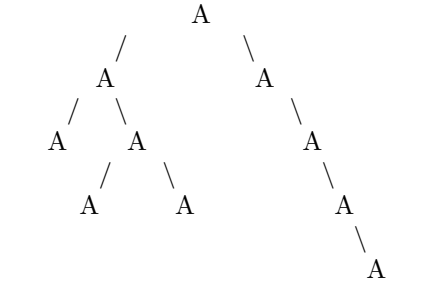
T= 510s : l'eolienne 64 s'active.



T = 1022s : l'eolienne 512 s'active.



T = 2046s: l'eolienne 1024 s'active.



au bout de 2046 secondes tout le parc est actif.

Implémentation

Afin de vous aider dans votre tâche, Nous vous demandons d'utiliser une des versions des arbres binaires du cours théorique.

Aussi, nous vous fournissons également un fichier *binary_eol.py* qui inclut la classe *Info* et auquel il faudrait ajouter l'implémentation de l'arbre du cours que vous avez choisie. La classe *Info* contient des informations supplémentaires sur chaque noeud. Elle est composée de :

- La profondeur du sous-arbre gauche.
- La profondeur du sous-arbre droit.
- Le temps nécessaire au signal pour atteindre l'éolienne à partir de la première éolienne active.
- Une variable booléenne pour vérifier si le sous-arbre actuel contient le nœud feuille initialement activé.

En d'autre termes :

- L'attribut *profondeurGauche* représente la hauteur maximale du sous-arbre gauche.
- L'attribut *profondeurDroite* représente la hauteur maximale du sous-arbre droit.
- La fonction *contient* est à True si le sous-arbre ayant pour racine le noeud actuel contient le noeud initialement activé.
- L'attribut *temps* qui représente le temps nécessaire pour que le signal atteigne le noeud actuel à partir de la feuille initialement active.

Nous vous demandons de compléter la classe *Solution* en implémentant: la méthode *calculer_temps* d'une manière récursive dans la classe Solution présente dans le fichier *binary_eol.py*. Cette méthode doit prendre trois paramètres:

- *noeud*: noeud actuel
- *info*: informations supplémentaires sur le noeud actuel
- *cible*: le noeud qui est activé

La classe Solution contient également un attribut *resultat* qui stockera le résultat du calcul.

Pour finir, la classe Solution contient la fonction *solve* qui fait appel à votre fonction *calculer_temps* avec les bons paramètres, vous ne devez pas la modifier.

Vous êtes libre de rajouter d'autres méthodes. Aucun paramètre supplémentaire n'est autorisé. Il est également conseillé d'implémenter des fonctions ou méthodes supplémentaires pour vérifier vos résultats.

Consignes Générales

Tout votre code doit être contenu dans un seul fichier *binary_eol.py* que vous compléterez à partir de celui disponible sur l'UV. Veuillez respecter le nom, le format des méthodes et le nom du fichier.

Un exemple de fichier est donné il sera passé en ligne de commande afin que vous puissiez tester votre code. Pour tester votre code, placez le fichier texte contenant l'arbre dans le même répertoire que votre fichier *binary_eol.py*, et lancez la commande:

```
python3 binary_eol.py arbre1.txt
```

Les informations de l'arbre se trouvent donc dans un fichier *arbreX.txt*. Chaque ligne dans ce fichier représente un noeud et ses deux enfants. L'arbre utilisé comme exemple ci-dessus est encodé dans le fichier *arbre1.txt* qui se trouve sur l'UV. Dans ce même fichier, si un noeud n'a pas de fils (gauche ou droit) la valeur None est assigné à celui-ci.

Nous vous conseillons de faire plusieurs tests en construisant des arbres dans des fichiers que vous aurez au préalablement créés.

Seront évalués la qualité de votre code, la mise en pratique de la matière vue en cours et les optimisations mises en place. Veillez à ce que votre code soit commenté de manière concise pour mettre en valeur ses fonctionnalités et les optimisations appliquées. Dans le cas où l'exécution de votre code en ligne de commande produit une erreur, une note nulle sera reportée pour la partie exécution. Veillez donc à tester toutes vos méthodes. Pour remettre votre projet sur l'UV, nous vous demandons de:

- Créer localement sur votre machine un répertoire de la forme NOM_Prenom (exemple : DUPONT_Jean) dans lequel vous mettez le *binary_eol.py* à soumettre.
- Compresser ce répertoire via un utilitaire d'archivage produisant un *.zip* (aucun autre format de compression n'est accepté).
- Soumettre le fichier archive *.zip*, et uniquement ce fichier, sur l'UV.

Le projet est à remettre pour le 03 avril 2022 à 23h59 sur l'UV. Tout manquement aux consignes ou retard sera sanctionné directement d'un **0/10**.