Projet Ls4 2014

Lancement du programme :
Pour lancer le programme, il suffit d'ouvrir le terminal, aller dans le dossier «projetLS4» et taper la commande : ./app.py

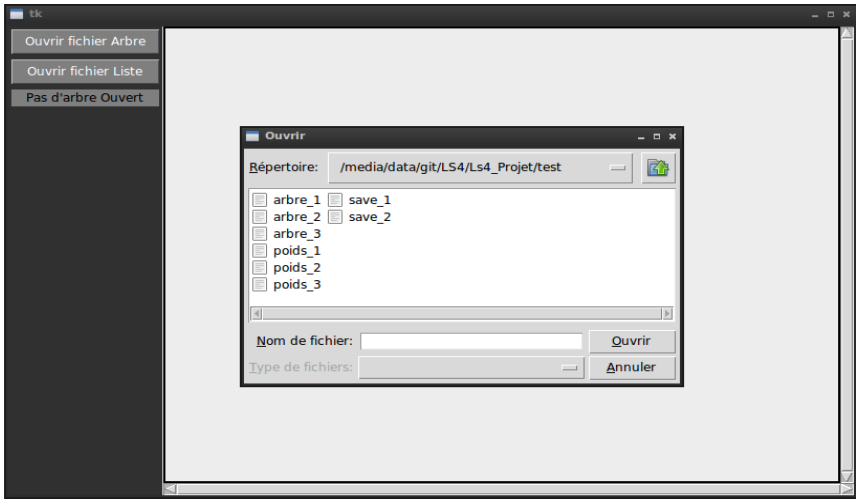
Utilisation du programme :
Une fois le programme lancé, une fenêtre apparaît (image 1) :

(image 1)

On pourra ouvrir deux types de fichiers :

Fichier arbre :

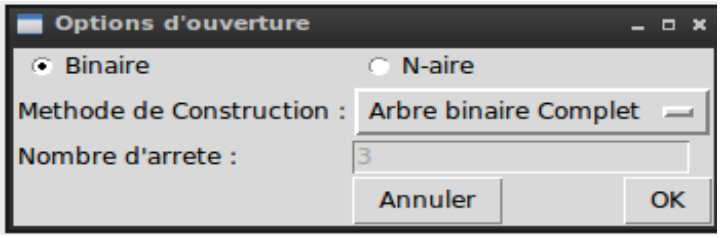
cette option nous invite à sélectionner un fichier contenant un arbre (si ce n'est pas un fichier arbre un message d'erreur apparaît).Ensuite l'arbre s'affiche sur la partie droite accompagné de certaines options su la partie gauche (voir image 2)



(Image 2)

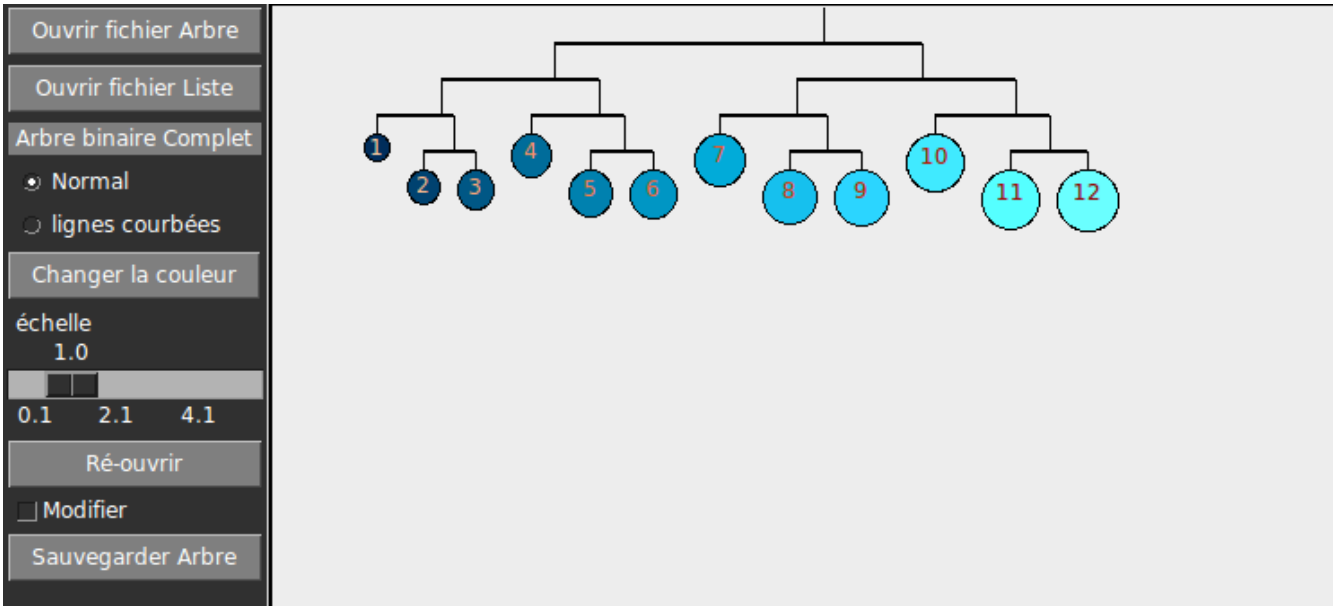
Fichier liste :

Elle permet de choisir un fichier liste (si ce n'est pas un fichier liste un message d'erreur apparaît) contenant une liste de poids. Une fois l'option sélectionné une boite de dialogue d'options d'ouverture s'affiche (voir image 3) .



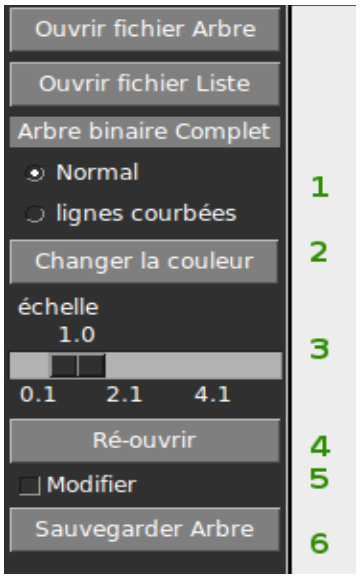
(image 3)

Ensuite on obtient une fenêtre (voir image 4).



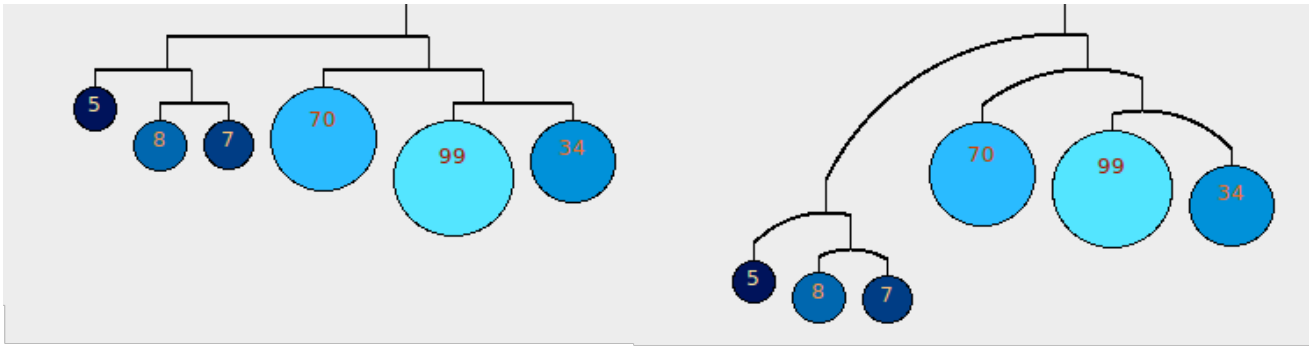
(Image 4)

Les fonctionnalités :



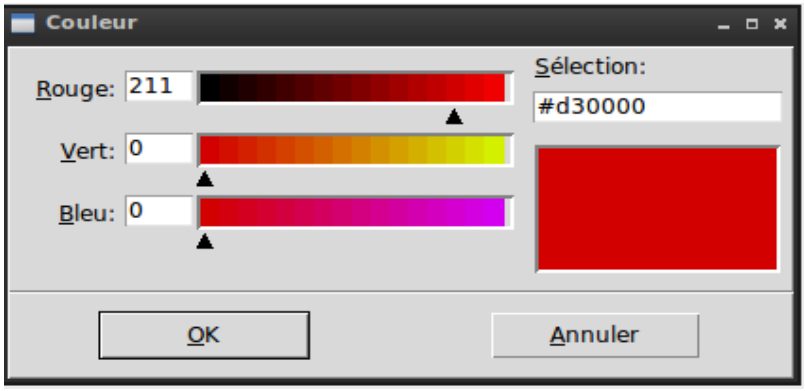
(Image 5)

(1) : Affichage de l'arbre : normal ou avec des lignes courbées .



(Image 6) *Affichage normal* *affichage lignes courbées*

(2) : Une palette de couleurs qui affiche l'arbre avec les poids coloriés en fonction de leurs poids : du plus clair (poids lourd) au plus clair (poids léger).

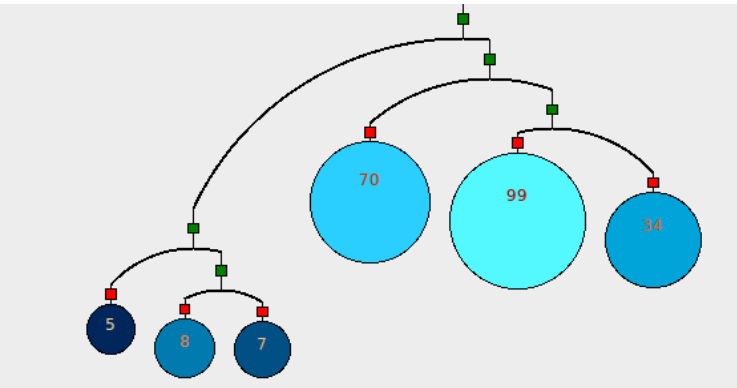


(Image 7)

(3) : Échelle : On peut zoomer ou dé-zoomer à votre convenance .

(4) : Ouvre la boite de dialogue (voir image 3) pour choisir l'algorithme de construction .

(5) : Si on active l'option, des petits points vont apparaître sur l'arbre



(Image 7)

vert pour modifier un arbre et rouge pour modifier un poids.

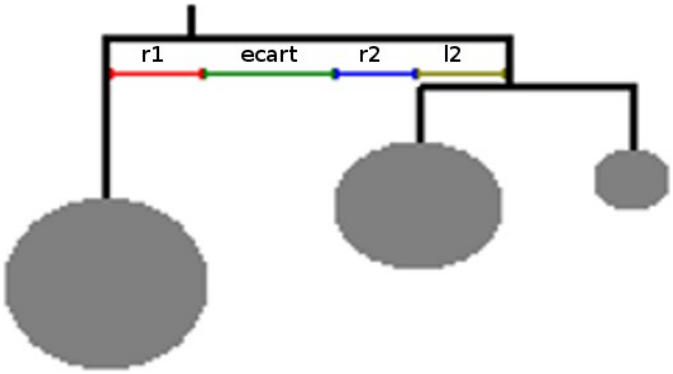
(6) : sauvegarder l'arbre dans un fichier

Description des algorithmes :

1) Tracer un arbre :

a)Normal :

La longueur de la tige est calculée comme sur le schéma1 :

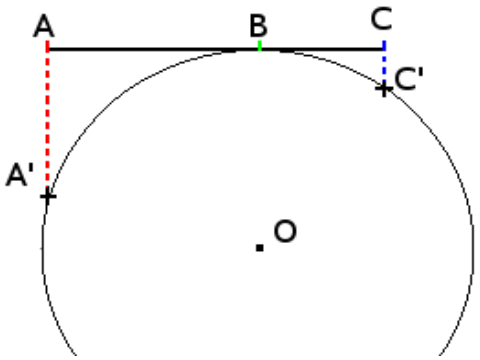


r1 = rayon du poids de gauche
écart = 10 pixel
r2 = rayon de poids au centre
l2 = la longueur dans le sous-arbre de droite allant du début de la tige jusqu'à son barycentre
La longueur de la tige est égal à $r1+écart+r2+l2$

Le diamètre des poids est $\sqrt[3]{POIDS}\times 15$ et ils sont ajoutés par récursion

b)Lignes courbées :

Voir schéma 2



B = point de barycentre
A = la position du sous-arbre de gauche
C = la position du sous-arbre de droite

Pour calculer les positions des points A' et C' on cherche les points d'intersection avec le cercle de centre O et de diamètre [AC]

2)Arbre binaire complet :

À partir d'une liste L.
Si L a plus d'un élément :
on cherche A= la plus grande puissance de deux inférieur ou égale à la taille de L;
on calcul le reste =taille de L-A
Si reste est supérieur à A :
La première liste contiendra les 2*A premiers éléments
Et la seconde liste les éléments restants
Sinon :
La première liste contiendra les A+reste premiers éléments
Et la seconde liste les éléments restants (a savoir A éléments)

Les branches gauche et droite sont construites récursivement avec les nouvelles listes.
Sinon.
l'élément correspond à notre poids.

3)Arbre alterné :

On prend le premier élément de la liste, on le place comme poids à gauche et on met le reste de la liste adroite, ensuite on applique la même stratégie sur cette dernière mais en plaçant cette fois ci le premier poids à droite et le reste à gauche ainsi de suite en alternant gauche et droit
exemple l={1,2,3,4} donne l'arbre [1,[[3,4],2]]

4) Arbre premier/non premier :

À partir de notre liste on crée deux liste une avec les nombres premiers et l'autre avec les nombres non-premiers .Ensuite pour chacune des listes on appelle la fonction des arbres binaires complets.

5)Arbre pair/impair :

À partir de notre liste on crée deux liste une avec les nombres pairs et l'autre avec les nombres impairs .Ensuite pour chacune des listes on appelle la fonction des arbres binaires complets.

6)Arbre n_aires :
À partir d'une liste L.
Si L a plus d'un élément :
on cherche A= la plus grande puissance de N inférieur ou égale à la taille de L;
on calcul le reste =taille de L-A
Si reste est supérieur à A :
La première liste contiendra les 2*A premiers éléments
Et la seconde liste les éléments restants
Sinon :
La première liste contiendra les A+reste premiers éléments
Et la seconde liste les éléments restants (a savoir A éléments)
etc.
Les N branches sont construites récursivement avec les nouvelles listes
Sinon.
l'élément correspond à notre poids.

7) Fort/Faible :

On tri la liste L et une nouvelle liste et construite en alternant le premier élément et dernier élément de L, ensuite on alterne dernier et premier(premier , dernier , dernier -1 , premier +1). Une fois le tri terminer on appelle la méthode de construction d'arbre binaire complet.

8) Croissant :

On trie la liste par ordre du poids ensuite on fait appel à la méthode d'arbre binaire complet.

Hypothèses retenues :

Dans nos méthode de construction, on est parti du principe que :

Le poids des tiges était nul ; donc notre calcul de barycentre ne prend en considération que la valeur des poids.

L'écart entre les poids est une constante = 10 afin d'éviter que les poids se chevauchent.

Dans la représentation des arbres avec lignes courbées, la construction a été réaliser en considérant que la taille de l'arc est égale à la taille de la tige.

comportements erronés ou inattendus :

À l'utilisation de notre programme vous pouvez rencontrer quelques comportements inattendus qui se produisent :

-Parfois il peut arriver que les tiges se touchent avec l'affichage de ligne courbée.

-Si le fichier (arbre ou poids) sélectionné comporte des erreurs non traitées, il peut occasionner des erreurs .

-La fonction eval() renvoie une erreur de mémoire quand les arbres sont très gros.

Liste des fonctionnalités non-implémentées :

Dans une première approche de notre projet, notre idée de départ comportait plus de fonctionnalités comme :

Un algorithmes qui calcul le barycentre en fonction des poids des tiges.

Un algorithme qui permet aux poids de se balancer.

Avoir plusieurs forme (autre que des boules) pour représenter les poids.

Amélioration possible :

On aurait aimé pouvoir améliorer notre projet en apportant une meilleure correction d'erreurs et en ajoutant d'autres fonctionnalités comme l'affichage en 3D des modules.

Bilan :

À la réalisation du projet nous avons pu développer nos compétences de programmation en python et apprendre à gérer les arbres ; construction d'arbre binaire et n_aires , algorithme de trie, algorithme de parcours.