<u>Ceci est le rapport du projet_algo_PMD_LE-GARS_GOHUGO.</u>

Il contient :

- Sources pour travaux au format de citation IEEE :
- Un état des annexes
- Un aperçu de notre travail de versionnage pour le travail collaboratif.
- Le pseudo-code de l'exercice 1 au format attendu.

SOURCES :

"Antisèches Python pour Biologiste Pressé." Available:

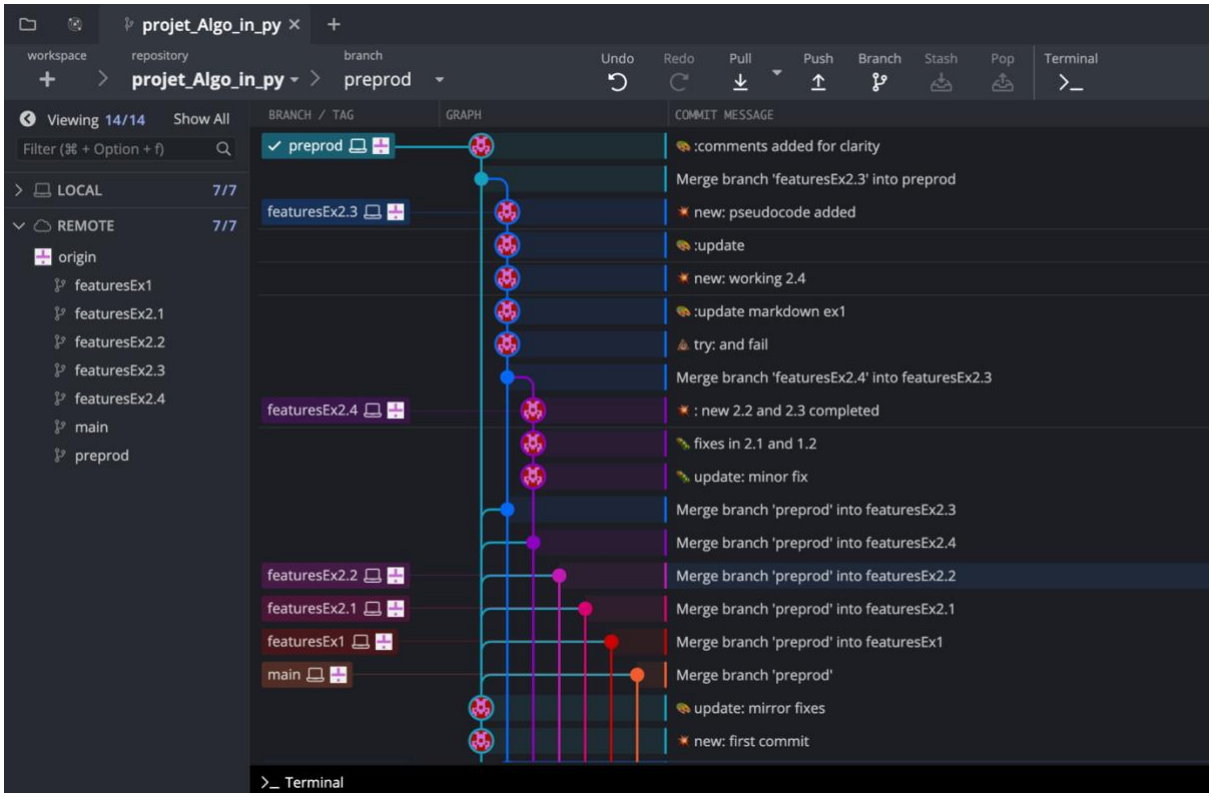
https://www.normalesup.org/~doulcier/teaching/python/00_antiseches.html

ANNEXES :

1. Un fichier Jupyter nommé '**projet.ipynb**' contenant l'ensemble du code ainsi que les explications et analyses, lorsque demandées, à l'intérieur du code ou en langage md. Fruit d'un travail collaboratif, certains commentaires et explications sont en anglais en accordances avec les bonnes pratiques de code.

2. Un fichier pdf reprenant le fichier 'projet.ipynb' appelé '**projet.pdf**'

VERSIONNAGE. (capture d'écran)



PSEUDOCODE EX 1.1 ET 1.2

VARIABLES

tab, *T* et *n* sont initialisés pour tous les algorithmes avec
LIRE *T,* ENTIER
tab ← [**POUR** ENTIER aléatoire (entre 0 et T) jusqu'à la borne T]
LIRE *n, ENTIER*

……

debut ← 0 ENTIER
fin ← *T* – 1 ENTIER
trouve ← FAUX BOULÉEN
start3 ← enregistre_temps FLOAT

DEBUT

TANT QUE *debut* ≤ *fin* && n ≥ *tab*[*debut*] && *n* ≤ *tab*[*fin*], **FAIRE**

probplace ← ‘FLOAT’ *debut* + (((*fin* - *debut*) * (*n* - *tab*[*debut*])) / (*tab*[*fin*] - *tab*[*debut*]))
probplace ← arrondi_vers_l’entier_inférieur(*probplace*) ENTIER

SI *debut* = *fin*, **ALORS**
SI *tab*[*debut*] = *n*, **ALORS**
trouve ← VRAI
quitter la boucle (« break »)
FIN SI
FIN SI

SI *tab*[*probplace*] = *n*, **ALORS**
trouve ← VRAI
quitter la boucle

SINON SI *tab*[*probplace*] < *n,* **ALORS**
debut ← *probplace* + 1
SINON
fin ← probplace - 1
FIN SI
FIN TANT QUE

end3 ← enregistre_temps FLOAT

SI *trouve* = VRAI, **ALORS**
AFFICHER "La valeur", *n*, "est à l'indice", *probplace*, "dans le tableau."
SINON
AFFICHER "Le nombre", *n*, "n'est pas dans le tableau."
FIN SI

AFFICHER "Temps de recherche par interpolation", end3 - start3
FIN

Complexité :

Le meilleur des cas se produit lorsque l'élément 'n' correspond à l'exacte milieu du tableau 'tab'. On trouve alors l'élément à la première itération de la boucle TANT QUE. Aussi, la complexité temporelle (temps d'exécution) dans une recherche constante (trouve du premier coup) est $\Theta(1)$.

Le pire des cas se produit lorsque l'élément n'est pas dans le tableau ou lorsqu'il est aux bornes de celui-ci. On itère alors un maximum de fois pour converger vers la position probable de l'élément. La complexité temporelle de l'algorithme est alors $\Theta(\log(\log n))$ où 'n' se confond avec la taille du tableau.

Variant de boucle :

Il s'agit de la différence entre *fin* et *début* car à chaque itération de la boucle, soit *début* est incrémenté (debut = probplace + 1), soit *fin* est décrémenté (fin = probplace – 1). On assure la fin de la boucle par la condition *début* > *fin*.

Invariant de boulce :

Il s'agit de l'élément qui reste vrai à chaque itération de la boucle. On peut considérer que n est l'invariant car il reste toujours entre les bornes du *tab*, *debut* et *fin*. On a donc toujours *debut* < n < *fin*.