Rapport TD4 Complexité algorythmique :

EX 2 :

fonction 1 :

complexité algorythmique = **$O(n^2)$ || O(n*x)**

La fonction analyse deux tableaux et incrémente un compteur a chauqe fois qu'ils ont une valeur en commun. Par conséquent, il va y avoir n appel au tableau 1 et x appel au taleau 2 . la complexité est donc de O(n*x). si ils font la meme taille : $O(n^2)$

fonction 2 :

complexité algorythmique = **O(n)**

La fonction additionne toutes les valeur de x en le décrémentant de 1 a chaque fois. Chaque itération du while est d'une complexité de 1, et s'éxexute x fois. La complexité algo est donc O(x), soit O(n).

fonction 3 :

complexité algorythmique = **O(1)**

La fonction retourne la valeur absolu de x. il n'y a aucune boucle, seulement une opération simple executer qu'une seul fois. Il y a donc une complxité algo de O(1).

EX 3 :

complexité algorythmique = **$O(n*x^3)$**

Pour calculer cette complexité, il va falloir additionner les différents bloc de cette fonction.

Boucle grades_number : complexité : **O(n)**
elle est executé grades_number fois. Appelons le n.
ce qui est dedans sera multiplié n fois.
Malloc = **O(1)**
boucle student number : complexité = **O(x)**
elle s'execute student number fois. Appelons le x
bublesort = **$O(x^2)$**
grade contient studentNumber element
boucle student number 2 : complexité =**$O(x^3)$**
s'execute egalement x fois
elle appele find_rank_students qui lui a une complexité e $O(x^2)$
Donc la complexité est de x*x*x
free = **O(1)**

En additionnant, on a donc le tout qui se répete n fois et le tout est : O(1) + O(x) + $O(x^2)$ +$O(x^3)$ + O(1) = $O(x^3)$.
Le retultat est donc : **$O(n*x^3)$**

EX 4 :

Complexité algorythmique =

Pour calculer cette complexité, il va falloir additionner les différents bloc de cette fonction.

Trie bulle = **$O(n^2)$**
La suite plus tard