**WATENDA SONGI ARSEL**

**ASIFIWE NYAMWOGA GLORY**

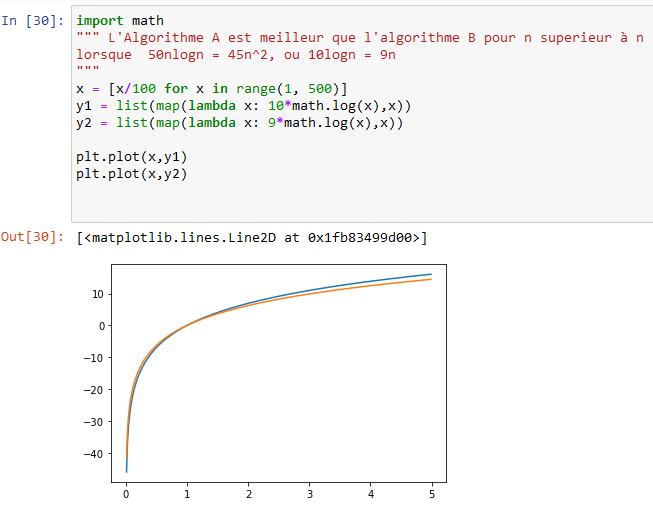
**MUKABI NGOMBO NELSON**

**2em G E I**

**TRAVAIL PRATIQUE D’ALGORITHMIQUE ET PROGRAMMATION**

1. On a A= 50nlogn et B=45n3

Pour que A soit meilleur que B, on doit écrire On a A=140n2 et B=29n3



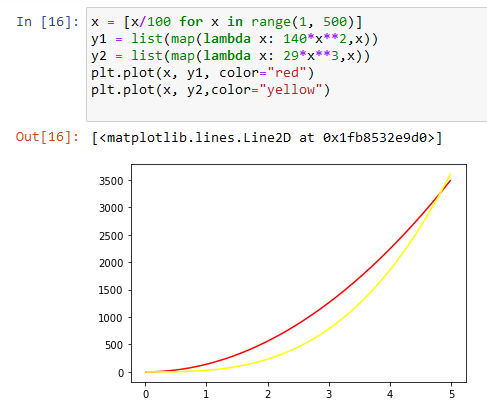
1. Ici aussi, on a A= 140n2 et B=29n3. Or, on veut que A soit meilleur que B donc, A>=B

=> 140n2 > 29n3

=>no<= 4.82757

À l’aide de python et son module

Matplotlib, nous avons fait la représentation graphique de nos deux fonctions :



1. A est O(f(n)). Si le temps d’exécution du pire des cas O(f(n)), il existe alors une constante c telle que cf(n)>= au pire cas pour n>no

Puisque le pire de cas est toujours supérieur ou égal à tout autre cas (le temps d’exécution du pire des cas >=(g(n))), cf(n)>=pire cas >= A

1. Si d(n)= O(f(n)) =>a\*d(n)=O(f(n)) pour a>0.

Si d(n) vaut O(f(n)) alors, il existe une constante c telle que d(n)<= c f(n) pour tout n>no.

On a alors ad(n)<= acf(n) = c’f(n).

La nouvelle constant sera donc “a” qui maintient toujours la condition originale de O qui sera vrai pour

1. Si d(n) équivaut à O(f(n)) et e(n) vaut O(g(n)), alors d(n) = cf(n) pour nF>nfo et e(n) < dg (n) pour n\_e>n\_e0 en conséquence, d(n)e (n) <= (cf (n)) (dg (n)) et n\_f\*n\_en\_f0\* 0

Ce qui signifie qu'il existe un nouveau n n\_f\*n\_e et et no' = n\_f0\*n\_e0, et un c’= c'd tel que d(n)e(n) <= e' (f(n)g(n)) pour n'>non', ce qui signifie que d(n)e(n) est O(f(n)g(n))

Et non fo\*n e0, et a c = c'd tel que \nd (n) e (n) <= c’ (f (n)g(n)) pour n'>non', ce qui signifie que d( n) e (n) est O(f(n) g (n))

1. Comme précédemment, si d(n) vaut o(f(n)) et e(n) vaut o(g(n)), alors d(n) <= cf (n) pour n\_f>n\_fo et e(n) <= dg (n) pour n\_e>n\_e0

Cela signifie que d(n) + e(n) <= (cf (n)) + (dg(n)) et

n > n\_f0+n\_e0

ce qui signifie qu'il existe un nouveau n'= n\_f+n\_e et et no = n\_f0+n\_e0, tel que :

d(n)e(n) <= cf (n) + dg (n) pour n>non' ; cependant, cela ne satisfait toujours pas la notation O.

On peut absorber c et d dans leurs fonctions telles que d(n) e(n) <= f(cn)+g(dn)

Pour absorber c, on note que n’> no/ c cn', donc n =n/ c, ce qui signifie que n'/ c

de même pour d, n’/ cd >= no/ cd

Il existe donc de nouvelles valeurs de no telles que

d(n) +e(n) <= ((n) +d(n)) pour n>= non, qui satisfait O(1(n) +d(n)) conditions

1. Le point clé ici est que ce n'est pas parce que quelque chose est O(n) que cela doit être cette fonction

Par exemple, f(n) 5 is O(n) est mathématiquement vrai, bien que ce soit une mauvaise forme de le dire

Par conséquent, si nous avons d(n) = n et e (n) = n avec f(n) = n et g(n) = n, alors nous vérifions d(n) <= C(f (n)) pour n>=0, et e(n) <= C2 (g(n)) pour n>=0

F(n)- g(n) = 0 et d(n)- e(n) = n

Il n'y a pas de valeur pour n>0 telle que 0>=n, ce qui signifie que d(n) -e(n) n'est pas 0(f(n)- g(n))

8. Comme précédemment, si d(n) vaut o(f(n)) et e(n) vaut o(g(n)), alors d(n) <= cf (n) pour n\_f>n\_fo et e(n) <= dg (n) pour n\_e>n\_e0

1. L'algorithme E est appelé par l’Algorithme D n fois

Par conséquent, le temps d'exécution le plus défavorable est O(D(n)) \*(0(i)), 0(n\*1) = O(n) d'après la description

1. La notation O signifie qu'il existe une constante c telle que f(n) < Cg(n)

Donc, si les algorithmes de Alphonse fonctionnent mieux que A (nlogn) et que l'algorithme

De Bob fonctionne mieux que B(n^2), nous pouvons résoudre la valeur où Anlogn = Bn^2, dont nous savons qu'elle est vraie Quand n=100. Cela signifie, (A/B) = (100) / (log (100)) = 15.05)

Cela signifie que le temps d'exécution de KIMBULU sur une seule itération

Est 15 fois plus lent mais comme il effectue globalement moins d'opération, et commence à mieux performé à des grandes valeurs de n