UNIVERSITE DE KINSHASA

FACULTE POLYTECHNIQUE

DEUXIEME GRADUAT

B.P 240 KINSHASA XI

TRAVAIL PRATIQUE D'ALGORITHMIQUE ET PROGRAMMATION

Dirigé par : Ass. Ir.

Fait par :

- MBONGA MABIALA GLOIRE (2GC)
- DZAPILI JOSEPH (2GEI)
- BOLOCHI KASIMA PIERRE (2GC)

ANNEE ACADEMIQUE 2021-2022

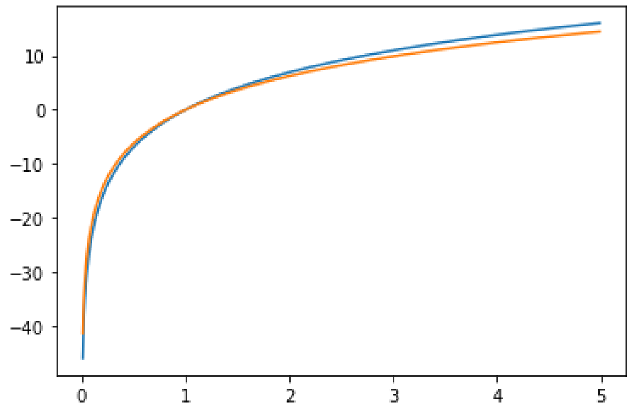
Question I

On a A= 50nlogn et B=$45n^3$
Pour que A soit meilleur que B, on doit écrire On a A=$140n^2$ et B=$29n^3$

```
In [30]: import math
         """ L'Algorithme A est meilleur que l'algorithme B pour n superieur à n
         lorsque  50nlogn = 45n^2, ou 10logn = 9n
         """
         x = [x/100 for x in range(1, 500)]
         y1 = list(map(lambda x: 10*math.log(x),x))
         y2 = list(map(lambda x: 9*math.log(x),x))

         plt.plot(x,y1)
         plt.plot(x,y2)

Out[30]: [<matplotlib.lines.Line2D at 0x1fb83499d00>]
```



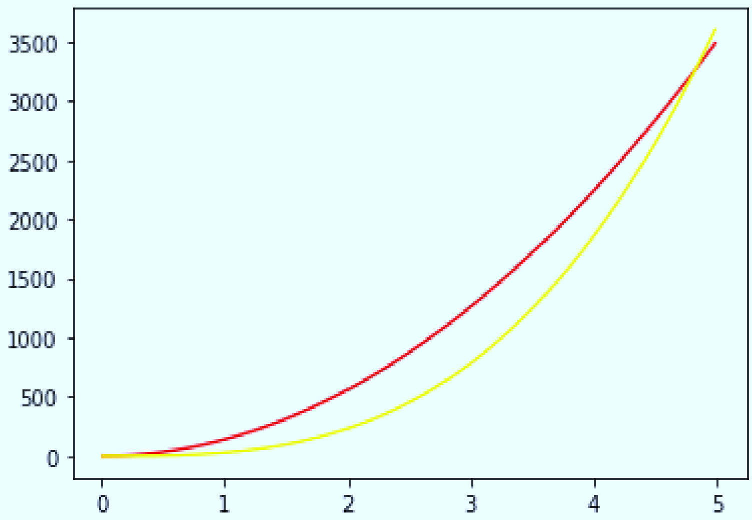
Question II

Ici aussi, on a A= $140n^2$ et B=$29n^3$. Or, on veut que A soit meilleur que B donc, $A \geq B$
→ $140n^2 > 29n^3$
→ no ≤ 4.82757

En utilisant, le module de python :
Matplotlib, ci dessous la représentation graphique de nos deux fonctions :

```
In [16]: x = [x/100 for x in range(1, 500)]
         y1 = list(map(lambda x: 140*x**2,x))
         y2 = list(map(lambda x: 29*x**3,x))
         plt.plot(x, y1, color="red")
         plt.plot(x, y2,color="yellow")

Out[16]: [<matplotlib.lines.Line2D at 0x1fb8532e9d0>]
```



Question III

Soit A == O(f(n)). Si le temps d’exécution du pire des cas O(f(n)) est n*, il existe alors une constante **c** telle que **c**f(n)≥ n*>n_o

Sachant le pire de cas est toujours supérieur ou égal à tout autre cas ; Du coup le temps d’exécution du pire des cas ≥(g(n))), **c**f(n) ≥ pire cas ≥ A

Question IV

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>n_o.

On a alors ad(n) ≤ acf(n) = c’f(n).

La nouvelle constante sera donc “a” qui maintient toujours la condition originale de O qui sera vrai pour tous les autres cas.

Question V

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) équivaut à O(f(n) g (n))

Question VI

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 tel que

d(n) +e(n) ≤ ((n) +d(n)) pour n≥ no, qui satisfait O(1(n) +d(n)) les conditions.

Question VII

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 = 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))

Question VIII

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

Question IX

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

Question X

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.