Complexité et Big O notation

October 22, 2019

1 Complexité et Big O notation

Dans ce document nous allons revoir les notions de complexité des algorithmes et en particulier la notion de Big O La notation Big O a pour but d'exprimer la complexité temporelle des algorithmes dans le pire scenario d'exécution (Worst Case scenario) et suit quelques principes de base:

- Les constantes sont ignorées

  $O(2n) = 2 * O(n) = O(n)$

- Les termes dominés sont ignorés

  $O(2n^2 + 5n + 50) = O(n^2)$

- Utilisez différentes variables pour différents inputs
- **Uniquement le pire cas d'exécution est consideré** (a.e. en cas de test logique)

1.0.1 Complexités fréquentes

Les complexités fréquentes sont les suivantes:

Constante $O(1)$ Cela veut dire que la complexité ne depend pas de la taille du vecteur en input. + les méthodes append et pop sur les listes en python

Logarithmique $O(log(n))$

- l'algorithme de recherche binaire

Linaire $O(n)$

- les boucles for et while

Quadratique, Cubique, etc... $O(n^2)$ **,** $O(n^3)$

- des boucles imbriquées ayant le même nombre d'itération
- l'algorithme de tri par selection à une complexité de $\frac{1}{2}(n^2 - n) = O(n^2)$

Exponentielle $O(2^n)$

- Fibonacci par recursion comme vu en exercice

1.0.2 Exemples

la boucle ci-dessous va exécuter n fois (le n ici est choisi arbitrairement et représente la taille du vecteur array) une opération de complexité $O(1)$, la complexité totale est donc $n * O(1) = O(n)$.

```
for number in array:
    array.pop()
```

Dans le prochain exemple nous avons deux vecteurs, si leur taille est respectivement de a et b alors la complexité sera de $O(a * b)$ et non $O(n^2)$, dans ce cas N ne represente rien.

```
def intersection_size(arrayA,arrayB):
    count = 0
    for a in arrayA:
        for b in arrayB:
            if a == b:
                count += 1
    return count
```

Les deux fonctions ci-dessous ont la même complexité de $O(n)$

```
def find_extrem_1(array):
    mini , maxi = (array[0], )*2
    for numbers in array:
        if numbers < mini:
            mini = numbers
    for numbers in array:
        if numbers > maxi:
            maxi = numbers
    return mini , maxi

def find_extrem_2(array):
    mini , maxi = (array[0], )*2
    for numbers in array:
        if numbers < mini:
            mini = numbers
        elif numbers > maxi:
            maxi = numbers
    return mini , maxi
```

Vidéos utiles

Algorithmes de tri

En dessous se trouve un lien vers une playlist sur les algorithmes de tri de **Micheal Sambol** sur youtube pour apprendre la différence entre les méthodes de tri.

Sorting playlist (https://www.youtube.com/playlist?list=PL9xmBV_5YoZOZSbGAXAPlq1BeUf4j20pl)

Complexité

[redacted]

[redacted]

(https://www.youtube.com/watch?v=__vX2sjlpXU&list=PL9xmBV_5YoZMxejjIyFHWa-4nKg6sdoIv)

In [ ]:

In [ ]: