



# Algo Invest & Trade



P07 - Emile MIATH

# SOMMAIRE



## 1. Graphique de notation Big O

## 2. Présentation de l'algorithme de force brute

A. Présentation de la méthode combinations

B. Présentation des étapes du programme

C. Avantages et inconvénients

## 3. Présentation de l'algorithme optimisé

A. Avantages et inconvénients

## 4. Comparaison avec les résultats de Sienna

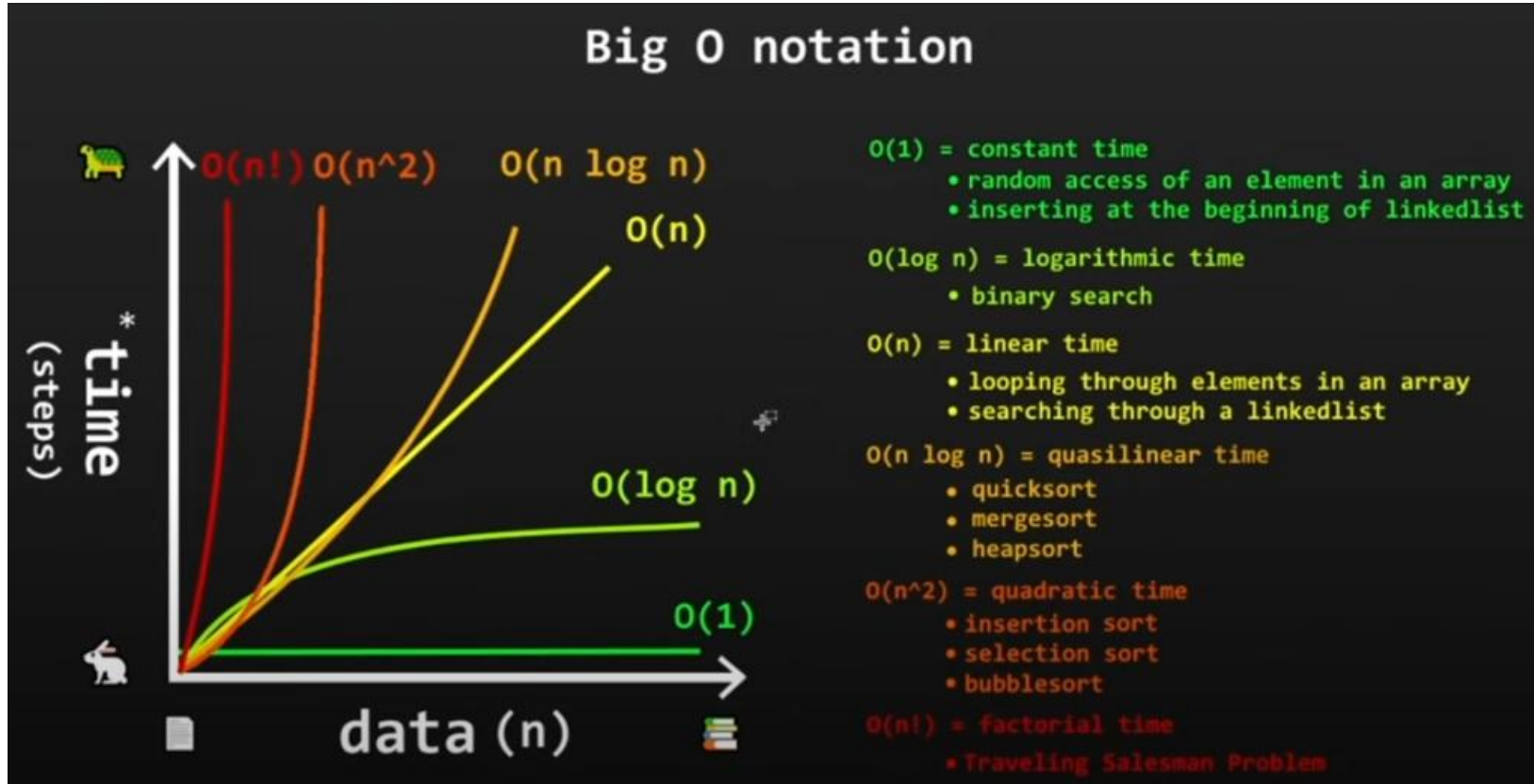
A. comparaison Dataset1

B. comparaison Dataset2

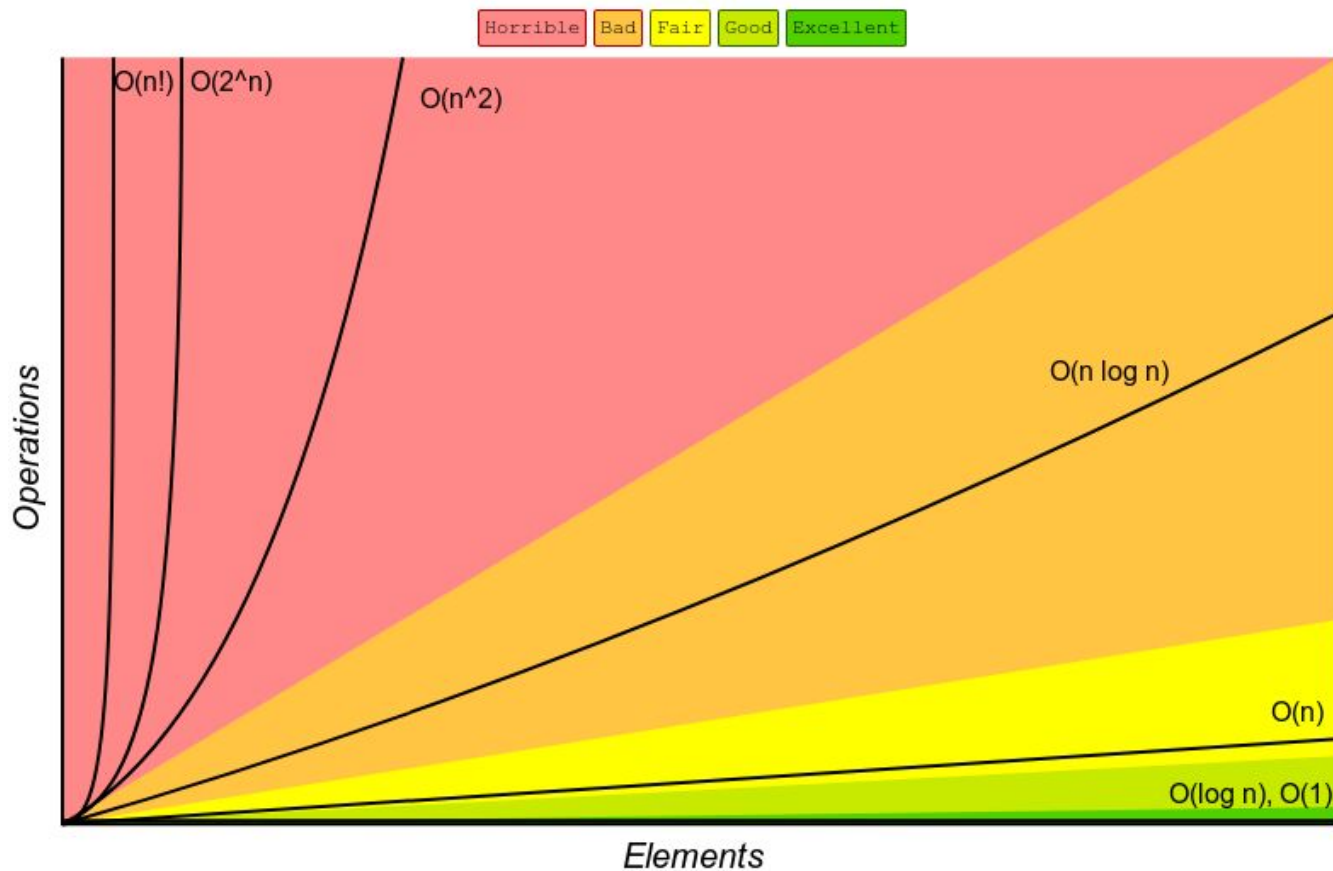
## 5. Conclusions

# 1. Les différentes notations Big O


Ce graphique nous montre les différentes courbes de complexités temporelle pour un algorithme donnée défini par leur notation Big O



Cet autre graphique nous montre que plus l'algorithme est pensé sous la forme  $2^n$  tout comme l'algorithme de force brute plus le temps de traitement sera long. Lors du traitement d'un plus grand nombre d'éléments il faudra préférer un algorithme optimisé de type  $O(n)$



## 2A. Présentation de la méthode combinations avec de l'algorithme de force brute



1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	131072
18	262144
19	524288
20	1048576

### Nombre de calculs effectués

On peut voir ici que si la colonne de gauche correspond à un nombre d'actions ou d'éléments, la colonne de droite représente ici le nombre de calcul de combinaison possible.

Ainsi, nous pouvons observer qu'en utilisant un algorithme de type  $2^n$  pour 20 actions, notre ordinateur devrait effectuer 1048576 calculs. A partir de cette observation nous pouvons nous rendre compte qu'en dépit de sa précision cette algorithme sera extrêmement lent pour plus de 20 actions et grignotera drastiquement la mémoire de notre pc.



Afin d'élaborer cet algorithme de force brute j'ai utilisé le module de combinaisons contenue dans la bibliothèque native python : Itertools.

Ce module permet notamment de récupérer toute les actions du dataset et tester toutes les combinaisons possibles les unes avec les autres jusqu'à atteindre la valeur maximale du wallet.

Il va créer des combinaisons sous forme de tuples qui seront implémentés dans une liste.

Ainsi, si nous n'avions que 4 actions à combiner, la première combinaison sera

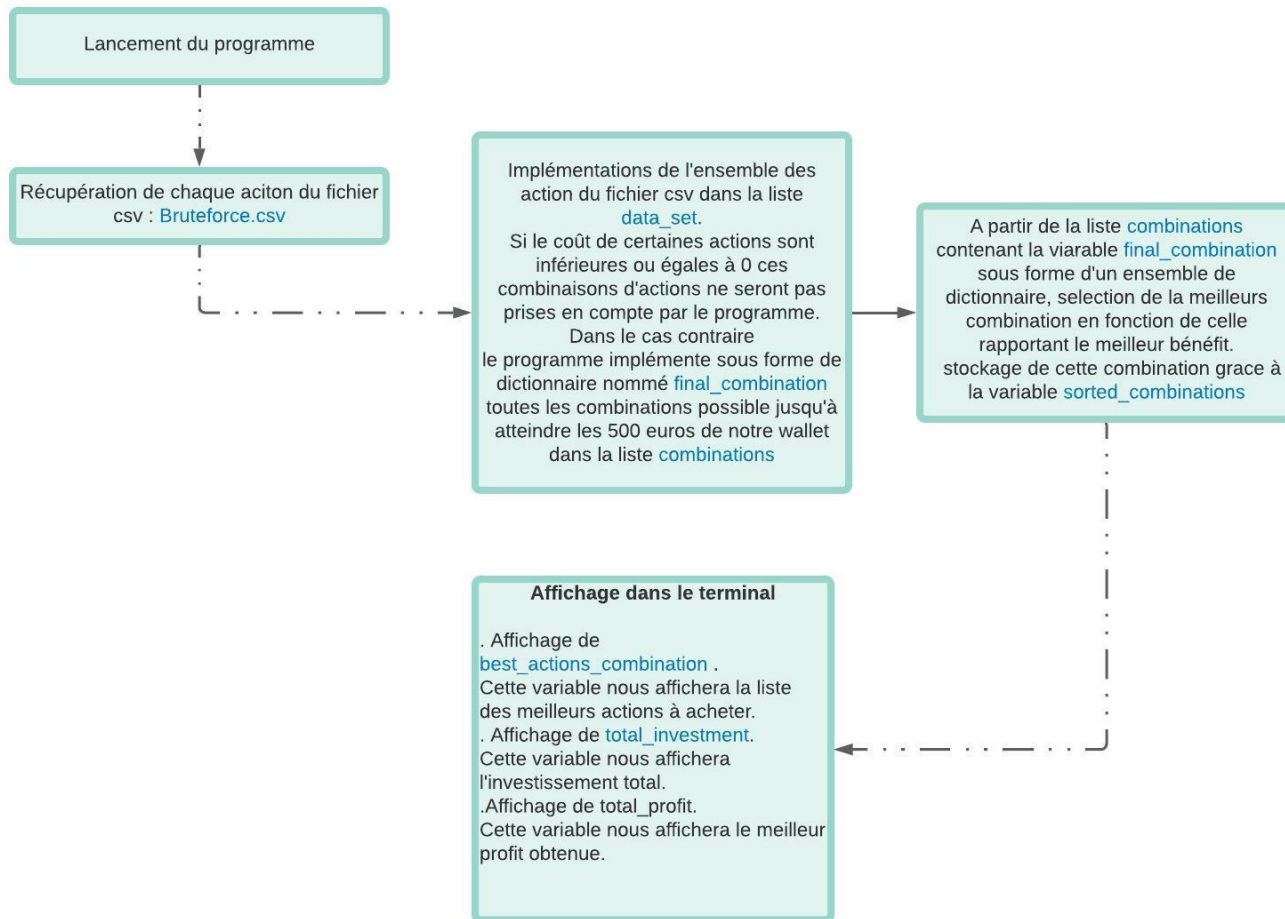
[(action-1,20,5), (action-2,30,10)] et la dernière

[(action-1,20,5), (action-2,30,10), (action-3,50,15), (action-4,70,20)]

il sera nécessaire de faire 16 calculs pour déterminer chaque combinaisons.

A chaque ajout d'une nouvelle action nous aurons alors un calcul de l'ordre de calculs ( $\text{calculs}^2$ )

## 2B. Étapes d'exécution du programme



## 2C. Avantages et inconvénients de l'algorithme de force brute



### Avantages:

- . Taux d'efficacité s'approchant des 100%
- . Marge d'erreur extrêmement réduite
- . Méthode parfaite du point de vue résultats pour un petit nombre d'éléments à traiter

### Inconvénients:

- . Complexité exponentielle
- . Énormément Chronophage et consomme énormément de mémoire



### 3. Algorithme Optimisé



L'algorithme optimisé est basé une solution d'algorithme dynamique.

Dans un tableau à deux dimensions contenant 0 dans chaque lignes et chaque colonnes .

Le nombre de ligne correspond au nombre d'actions présentes dans le dataset tandis que le nombre de colonnes correspond à chaque unité du budget maximum investi.

Il parcourt alors les cellules de cette matrice une par une, une seule et unique fois.

Son exploitation permet alors de déterminer la meilleur liste d'actions à acheter.

### 3A. Avantages et inconvénients de l'algorithme optimisé



#### Avantages:

- . Exécutions très rapide en moins d'une seconde.
- . Tri effectué selon plusieurs paramètres et comparaison pour une sortie optimale

#### Inconvénients:

- . Marge d'erreur potentielle par rapport à la méthode de force brute

## 4A. Algorithme optimisé : Comparaison des résultats avec Sienna

### Dataset 1

For the better investment you need to buy this list of actions :

-----  
Share-DBUJ  
Share-KMTG  
Share-GHIZ  
Share-CYYC  
Share-NHWA  
Share-UEZB  
Share-IQMC  
Share-LPDM  
Share-MTLR  
Share-GTQK  
Share-FKJW  
Share-EVUW  
Share-QLMK  
Share-CGJM  
Share-WPLI  
Share-LGWG  
Share-ZSDE  
Share-LOKP  
Share-SKKC  
Share-STKT  
Share-QQTU  
Share-PBXL  
Share-KGQI  
Share-CBNY  
Share-XJMO  
Share-LRBZ  
Share-EMOV  
Share-IFCP  
-----

For a total investment of 500€

You will win : 199.93€

That means that your return on investment is : 39.99 %

Program executed in 0.41 seconds

Sienna bought:

Share-GRUT

Total cost: 498.76€

Total return: 196.61€

## 4B. Algorithme optimisé : Comparaison des résultats avec Sienna

### Dataset 2

For the better investment you need to buy this list of actions :

-----  
Share-IXCI  
Share-FWBE  
Share-ZOFA  
Share-PLLK  
Share-MEQV  
Share-LXZU  
Share-PATS  
Share-SCWM  
Share-ZLMC  
Share-VCXT  
Share-NDKR  
Share-ALIY  
Share-JWGF  
Share-FCHD  
Share-LKSD  
Share-JGTW  
Share-VCAX  
Share-LFXB  
Share-DWSK  
Share-UPCV  
Share-DYVD  
Share-XQII  
Share-OAVO  
Share-ROOM  
Share-YCGH  
-----

For a total investment of 500€

You will win : 199.04€

That means that your return on investment is : 39.81 %

Program executed in 0.21 seconds

Sienna bought:

Share-ECAQ 3166  
Share-IXCI 2632  
Share-FWBE 1830  
Share-ZOFA 2532  
Share-PLLK 1994  
Share-YFVZ 2255  
Share-ANFX 3854  
Share-PATS 2770  
Share-NDKR 3306  
Share-ALIY 2908  
Share-JWGF 4869  
Share-JGTW 3529  
Share-FAPS 3257  
Share-VCAX 2742  
Share-LFXB 1483  
Share-DWSK 2949  
Share-XQII 1342  
Share-ROOM 1506

Total cost: 489.24€

Profit: 193.78€



Pour  $n$  = nombre d'  
éléments à analyser

### Brute Force

- . Complexité de temps  $O(2^n)$
- . Complexité de mémoire  $O(2^n)$
- . Exponentielle

### Optimisé

- . Complexité de temps  $O(n)$
- . Complexité de mémoire  $O(n)$
- . linéaire