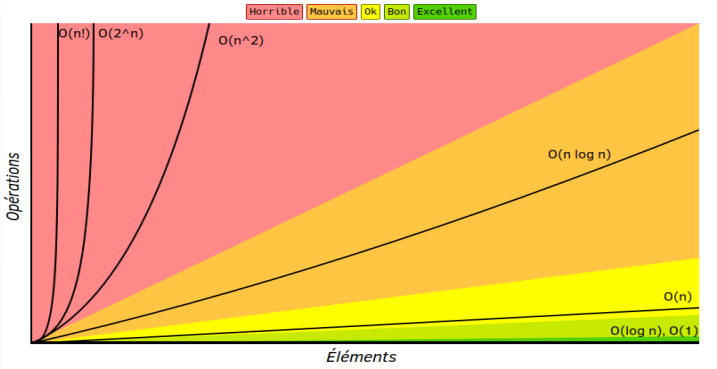
Projet 7 –
SOLUTION OPTIMISÉE

1. Analyse de l’algorithme de force brute

Analyse

- L'algorithme de force brute
  - Nombre de combinaison de plus en plus importante
  - Temps de traitement long



2. Processus de reflexion de la solution

Reflexion de l'algorithme force brute

- Dans le premier algorithme, le mode de fonctionnement est le suivant :
  - On teste d'abord toutes les combinaisons possible,
    (1)
    (1,2)(1,3)(1,4)(1,5)(1,6)(1,7)...
    (1,2,3)(1,2,4)(1,2,5)(1,2,6)(1,2,7)...
    ...
  - On calcule ensuite le coût et le profit de chaque combinaison.
  - On supprime les combinaisons > 500€
  - On choisit la combinaison avec les bénéfices les plus élevés.

Actions #	Coût par action (en euros)	Bénéfice (après 2 ans)
Action-1	20	0,05
Action-2	30	0,1
Action-3	50	0,15
Action-4	70	0,2
Action-5	60	0,17
Action-6	80	0,25
Action-7	22	0,07
Action-8	26	0,11
Action-9	48	0,13
Action-10	34	0,27
Action-11	42	0,17
Action-12	110	0,09
Action-13	38	0,23
Action-14	14	0,01
Action-15	18	0,03
Action-16	8	0,08
Action-17	4	0,12
Action-18	10	0,14
Action-19	24	0,21
Action-20	114	0,18

Reflexion de l'algorithme optimal

- Dans ce nouvel algorithme, le but est de fonctionner de manière à trier les données selon leurs bénéfices afin d'arriver à une solution optimisée d'un point de vue temps de traitement et proposition d'achat des actions.

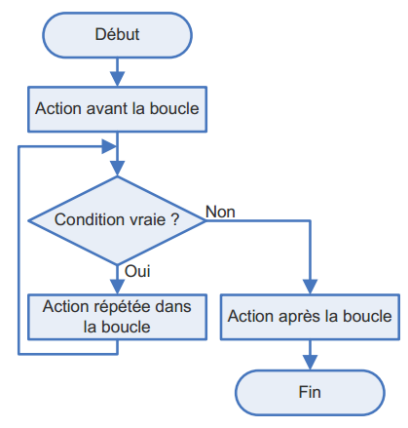
Actions #	Coût par action (en euros)	Bénéfice (après 2 ans)
Action-1	20	0,05
Action-2	30	0,1
Action-3	50	0,15
Action-4	70	0,2
Action-5	60	0,17
Action-6	80	0,25
Action-7	22	0,07
Action-8	26	0,11
Action-9	48	0,13
Action-10	34	0,27
Action-11	42	0,17
Action-12	110	0,09
Action-13	38	0,23
Action-14	14	0,01
Action-15	18	0,03
Action-16	8	0,08
Action-17	4	0,12
Action-18	10	0,14
Action-19	24	0,21
Action-20	114	0,18

Actions #	Coût par action (en euros)	Bénéfice (après 2 ans)
Action-10	34	0,27
Action-6	80	0,25
Action-13	38	0,23
Action-19	24	0,21
Action-4	70	0,2
Action-20	114	0,18
Action-5	60	0,17
Action-11	42	0,17
Action-3	50	0,15
Action-18	10	0,14
Action-9	48	0,13
Action-17	4	0,12
Action-8	26	0,11
Action-2	30	0,1
Action-12	110	0,09
Action-16	8	0,08
Action-7	22	0,07
Action-1	20	0,05
Action-15	18	0,03
Action-14	14	0,01

3. L’algorithme et ses limites

Type d'algorithme

- Type Boucle: ou algorithme de répétition, il s'éxecutera jusqu'à ce que la limite de budget soit atteinte.



Limites

- Dans le cas où une action serait proche de la limite d'argent que nous avons, la solution pourrait laisser passer celle-ci et donc ne pas être forcément la plus optimisée.
- Il est possible de régler ce problème en affichant les valeurs les plus élevées à l'utilisateur si nécessaire, mais dans notre cas, la solution semble être optimisée!

4. Analyse des performances

Comparaison

```
After 807943 combinations under your rules, the best Stocks to buy are :
Action-4
Action-5
Action-6
Action-8
Action-10
Action-11
Action-13
Action-18Â
Action-19
Action-20
It's worth 99.08 after 2 years!

real    0m2,579s
user    0m0,000s
sys     0m0,000s
```

Notation Big-O : $O(2^n)$

```
The best Stocks to buy are :
Action-10
Action-6
Action-13
Action-19
Action-4
Action-20
Action-5
Action-11
Action-18Â
Action-17
Action-16
Action-14
For 498.0 it's worth 0.97 after 2 years!

real    0m0,181s
user    0m0,000s
sys     0m0,047s
```

Notation Big-O : $O(n)$

Comparaison

def openFile:
affectation : 1
affectation : 1
itération : au plus n
affectation + multiplication : 2
affectation : 1
TOTAL : n+5

def powerset:
itération : au plus n
TOTAL : n

getCombination:
itération : au plus n
TOTAL : n

valeurMax:
affectation : 1
itération : au plus n
affectation : 1
itération : au plus n
affectation : 1
TOTAL : n^2 + 3

calculateProfit:
itération : au plus n
affectation : 1
itération : au plus n
affectation : 1
affectation : 1
TOTAL : n^2 + 3

calculateFinalProfit:
affectation : 1
itération : au plus n
TOTAL : 1 + n

main:
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
Total : 12

TOTAL : n+5 + n + n + n^2 + 3 + n^2 + 3 + 1 + n + 12
Total : 2n^2 + 4n + 24

Complexité temporelle : $2n^2 + 4n + 24$

def openFile:
affectation : 1
affectation : 1
itération : au plus n
affectation + multiplication : 2
affectation : 1
TOTAL : n+5

def findStocks:
itération : au plus n
affectation : 1
affectation : 1
affectation : 1
TOTAL : n + 3

getKeysByValue:
affectation : 1
itération : au plus n
TOTAL : n + 1

printSolution:
itération : au plus n
TOTAL : n

main:
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
affectation : 1
itération : au plus n
itération : au plus n
itération : au plus n
Total : 3n + 7

TOTAL : n + 5 + n + 3 + n + 1 + n + 3n + 7
Total : 7n + 16

Complexité temporelle : 7n + 16

Comparaison

def openFile:	def powerset:	def valeurMax	
long = 8	int [] = 4N + 24	int = 4	
int = 4		int = 4	
float = 4		int = 4	
float = 4		int = 4	
char [][] = 2N			
2n + 20	4N + 24	16	
def calculateProfit:	**def calculateFinalProfit:**	**main:**	**TOTAL:**
int = 4	int = 4	long = 8	52N + 276
int = 4	int = 4	long = 8	
int = 4		char[] = 2N + 24	
int = 4		double[] = 8N + 24	
		double[] = 8N + 24	
		double[] = 8N + 24	
16	8	double[] = 8N + 24	
		int = 4	
		int = 4	
		double[] = 8N + 24	
		int[] = 4N + 24	
		46N + 192	

Analyse de la mémoire : 52N + 276

def openFile:	def findStocks:	
long = 8	int = 4	
int = 4	float = 4	
float = 4	float = 4	
float = 4		
char [][] = 2N		
2n + 20	12	
def getKeysByValue	**main:**	**TOTAL:**
double[] = 8N + 24	long = 8	34N + 84
	long = 8	
8N + 24	double[] = 8N + 24	
	int[] = 4N + 24	
	double[] = 8N + 24	
	int = 4	
	int[] = 4N + 24	
	int = 4	
	int = 4	
	int = 4	
	24N + 128	

Analyse de la mémoire : 34N + 84

Résultats des algorithmes

Dataset 1:

```
Sienna bought:

Share-GRUT

Total cost: 498.76â,¬
Total return: 196.61â,¬
```

Dataset 2:

```
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: 489.24â,¬
Profit: 193.78â,¬
```

Dataset 1:

```
The best Stocks to buy are :
Share-XJMO
Share-MTLR
Share-KMTG
Share-LRBZ
Share-GTQK
Share-WPLI
Share-GIAJ
Share-GHIZ
Share-IFCP
Share-ZSDE
Share-FKJW
Share-NHWA
Share-LPDM
Share-QQTU
Share-USSR
Share-EMOV
Share-LGWG
Share-SKKC
Share-QLMK
Share-UEZB
Share-CBNY
Share-CGJM
Share-EVUW
Share-FHZN
Share-MLGM
For 499.94 it's worth 198.49 after 2 years!

real    0m0,239s
user    0m0,000s
sys     0m0,061s
```

Dataset 2:

```
The best Stocks to buy are :
Share-PATS
Share-JWGF
Share-ALIY
Share-NDKR
Share-PLLK
Share-FWBE
Share-LFXB
Share-ZOFA
Share-ANFX
Share-FAPS
Share-LXZU
Share-XQII
Share-ECAQ
Share-JGTW
Share-IXCI
Share-DWSK
Share-ROOM
Share-VCXT
Share-YFVZ
Share-OCKK
Share-JMLZ
Share-DYVD
For 499.98 it's worth 197.75 after 2 years!

real    0m0,219s
user    0m0,000s
sys     0m0,047s
```