AlgoInvest&Trade

Les algorithmes au service de la stratégie d'investissement

Sommaire

1. Présentation des algorithmes :

- Algorithme de force brute,
- Algorithme optimisé (sac à dos),
- Présentation de l'algorithmes optimisés,
- Explication du choix parmi deux algorithmes optimisés,
- Analyse de performance.

2. Rapport d'exploration :

- Comparaison Data set 1,
- Bilan,
- Comparaison Dataset 2,
- Bilan.

1. Présentation des algorithmes

Plusieurs algorithmes sont étudiés, offrant chacun des avantages et limites différents.

Algorithme de force brute

- Principe :

  Parcourt toutes les possibilités et sélectionne la meilleure.

- Performances :

  Big O : O(2^n) | n = nombre d'éléments

- Limites :

  Le temps de traitement sera en fonction du nombre d'éléments (croissance exponentielle)

Algorithme optimisée (sac à dos)

- Principe :

  Création d'une matrice qui stocke les résultats transitoires. L'algorithme va calculer une seule fois les possibilités et fera des comparaisons pour retenir la meilleure.

- Preformance :

  Big O : O(n*w) | n = nombre d'éléments , w = capacité

- Limites :

  Utilise de la mémoire en fonction du nombre d'items et de capacité.
  Stockage de résultats transitoires (matrice) pour obtenir par comparaison la solution définitive.

Présentation de l’algorithme optimisé

```
FONCTION algo_Dynamique(invest, shares_list)
    Variables
        invest ← 500
        n ← nombre d'action
        cost ← []
        profit ← []
    Début
        POUR shares DANS shares_list :
            cost ← la première valeur de shares_list
            profit ← la deuxième valeur de shares_liste
        FIN_POUR

        Variables
        matrice ← []
            POUR x JUSQU'A invest+1:
                x ← 0
            FIN_POUR
            POUR x JUSQU'A le n+1:
                x ← 0
            FIN_POUR
        POUR i ALLANT DE n A n+1
            POUR w ALLANT DE invest A invest+1
                SI le dernier cout de l'action est inferieur à la capacité de l'investissement
                    matrice[i][w] ← prendre le maximum entre (l'element courant + la solution optimisé de la ligne du
                                    dessus, moins le cout de l'element de ligne d'avant) ET la solution optimisé de la
                                    ligne du dessus
                SINON
                conserver la solution optimisé de la ligne du dessus
                FIN_SI
            FIN_POUR
        FIN_POUR

     Variables
     shares_best ← []
    TANT_QUE invest SUPERIEUR OU EGALE A 0 ET n SUPERIEUR OU EGALE A 0
        Variable
        e ← shares_list[n-1]
        SI la valeur de l'invest de l'element EST STRICTEMENT EGAL A la valeur de l'invest de l'element[n-1] - cost[n-1] + profit[n-1] :
            shares_best ← e
            invest ← - cost[n-1]
        FIN_SI
        n ← - 1
    FIN_TANT_QUE
    RETOURNE shares_liste
    FIN
```

Explication du choix parmi deux algorithmes optimisés

Explication

Pour calculer notre algorithme devons faire avoir les données prix en entier :

- Algorithme 1 : nous multiplions la valeur et la capacité par 100 afin d'obtenir un nombre entier. Nous pouvons conserver les arrondis en redivisant par 100 en sortie. Ainsi la précision du résultats est conservée Toutefois, la matrice voit sont nombre de calcule augmenter car nous avons multiplié l'investissement (w) par 100. La performance est 0(1 000 000)
- Algorithme 2 : les données initiales des prix sont enregistrées en entier. La performance est 0(10 000) Mais en sortie nous aurons les données en entier et nous perdons de la précision en sortie.

BILAN

- Ici nous présentons deux algorithmes optimisés car ils ne présentent pas les mêmes avantages.
- L'un va être extrêmement rapide (algo 2) mais moins précis (résultat à l'entier).
- L'autre sera plus précis avec un résultat en décimal (algo 1) avec un temps d'application plus rapide que la force brute mais plus lent que l'algo 2.
- En fonction de vos besoins et des données vous pouvez passer de l'un à l'autre.

Analyse de performance

Brute Force

Invest (w)	Action (n)	O(2^n)	Time seconds
500	20	1 048 576	3.74

- refait les calculs
- Très lent
- Bon pour un problème avec un petit nombre de données à traiter

Optimisé 1

Invest (w)	Action (n)	O(n*w)	Time seconds
5000	20	1 000 000	0.69

- Parcours les résultats déjà obtenu et fait une comparaison
- D'où une réponse plus rapide
- Plus lent que l'algorithme 2 mais gagne en précision

Optimisé 2

Invest (w)	Action (n)	O(n*w)	Time seconds
500	20	10 000	0.05

- Parcours les résultats déjà obtenu et fait une comparaison
- D'où une réponse plus rapide
- Plus rapide mais perd en précision sur le retour

2. Rapport d’exploitation :

Comparaison des l’algorithmes et les choix de Sienna

Rapport d'exploitation :

dataset 1

Sienna

```
Sienna bought:

Share-GRUT

Total cost: 498.76€
Total return: 196.61€
```

Optimisé 1

```
The most profitable 22 shares are :

Share-KMTG | Share-GTQK | Share-QQTU |
Share-GHIZ | Share-FKJW | Share-GIAJ |
Share-NHWA | Share-MLGM | Share-XJMO |
Share-UEZB | Share-QLMK | Share-LRBZ |
Share-LPDM | Share-WPLI | Share-KZBL |
Share-MTLR | Share-LGWG | Share-EMOV |
Share-USSR | Share-ZSDE | Share-IFCP |
           | Share-SKKC |

Total cost :  499.95 €.
Profit after 2 years : + 198.55 €.

Time elapsed :  21.6759192943573 seconds
```

Optimisé 2

```
The most profitable 33 shares are :

Share-CUSU | Share-KFOG | Share-SKKC | Share-EMOV |
Share-KMTG | Share-UEZE | Share-TXHQ | Share-IFCP |
Share-KXOH | Share-YPGY | Share-PBXL | Share-RUFN |
Share-DXOW | Share-EVUW | Share-GIAJ | Share-CFOZ |
Share-UEZB | Share-QQGZ | Share-CBNY |
Share-LPDM | Share-QSQG | Share-LSZT |
Share-MTLR | Share-QLMK | Share-SNKS |
Share-USSR | Share-WPLI | Share-XJMO |
Share-GTQK | Share-ZSDE | Share-LRBZ |
           | Share-ZNKU | Share-KZBL |

Total cost :  500 €
Profit after 2 years : + 204.28 €

Time elapsed :  0.23773550987243652 seconds
```

Rapport d’exploitation :

dataset 1

Bilan

- Sienna obtient un profit moins avantageux que les deux algorithmes proposés.
- Sienna propose qu’une seule action contre 22 et 33 pour les algorithmes présentés.
- L’algorithme 1 est plus précis (centième). Mais met plus de temps à retourner les réponses (choix de conserver la précision sur le coût de l’investissement en nombre décimal).
- Quant à l’algorithme 2 voit des profits supérieur. La réponse est extrêmement rapide, mais le coût de l’investissement est en entier et nous perdons la précision avec un arrondis.

Rapport d'exploitation :

dataset 2

Sienna

```
Sienna bought:
Share-ECAQ  Share-NDKR  Share-XQII
Share-IXCI  Share-ALIY  Share-ROOM
Share-FWBE  Share-JWGF
Share-ZOFA  Share-JGTW
Share-PLLK  Share-FAPS
Share-YFVZ  Share-VCAX
Share-ANFX  Share-LFXB
Share-PATS  Share-DWSK

Total cost: 489.24€
Profit: 193.78€
```

Optimisé 1

```
The most profitable 20 shares are :

Share-ECAQ  Share-YFVZ  Share-JGTW
Share-IXCI  Share-ANFX  Share-FAPS
Share-FWBE  Share-PATS  Share-VCAX
Share-ZOFA  Share-SCWM  Share-LFXB
Share-PLLK  Share-NDKR  Share-DWSK
Share-LXZU  Share-ALIY  Share-XQII
            Share-JWGF  Share-ROOM

Total cost :  499.9 €.
Profit after 2 years : + 197.96 €.

Time elapsed :  11.03798794746399 seconds
```

Optimisé 2

```
The most profitable 23 shares are
Share-ECAQ  Share-ANFX  Share-IJFT
Share-IXCI  Share-PATS  Share-VQQX
Share-FWBE  Share-NDKR  Share-LFXB
Share-ZOFA  Share-ALIY  Share-DWSK
Share-PLLK  Share-JWGF  Share-BMHD
Share-MEQV  Share-FAKH  Share-XQII
Share-LXZU  Share-FAPS  Share-GEBJ
Share-YFVZ  Share-ZKSN

Total cost :  500 €
Profit after 2 years : + 202.5 €

Time elapsed :  0.14998102188110352 seconds
```

Rapport d’exploitation :

dataset 2

Bilan

- Sienna obtient un profit moins avantageux que les deux algorithmes proposés.
- L’algorithme 1 est plus précis (centième). Mais met plus de temps à retourné les réponses (choix de conserver la précision sur le coût de l’investissement en nombre décimal).
- Quant à l’algorithme 2 voit des profits supérieur mais le coût de l’investissement est un arrondis.

Merci pour votre écoute.