

Page 1 : Le Problème à résoudre

⌚ La Situation

Imaginez que vous avez **500€** à investir dans des actions.

Il y a **20 actions** disponibles :

- Certaines coûtent 10€
- D'autres coûtent 100€
- Toutes rapportent un certain pourcentage après 2 ans

❓ La Question

Comment choisir les meilleures actions pour gagner le maximum d'argent ?

💡 La Réponse

Utiliser des **algorithmes** (= des recettes informatiques) pour trouver la meilleure combinaison automatiquement !

Exemple Concret :

Vous avez 100€ et 3 actions :

Action A : 40€, rapporte 20% → 8€ de gain
Action B : 50€, rapporte 15% → 7.50€ de gain
Action C : 30€, rapporte 10% → 3€ de gain

Quelle combinaison choisir ?

Réponse : A + C = 70€ investis, 11€ de gain !

Page 2 : Les 3 Méthodes Testées

Nous avons testé **3 façons différentes** de résoudre le problème :

[1] Méthode Force Brute (Le Perfectionniste)

Comment ça marche :

- Teste TOUTES les combinaisons possibles
- Garde la meilleure

Analogie : C'est comme essayer TOUS les vêtements d'un magasin pour trouver la tenue parfaite.

Avantage : Trouve LA meilleure solution

Problème : Très très lent

[2] Méthode Glouton (Le Rapide)

Comment ça marche :

- Prend les actions les plus rentables en premier
- Continue jusqu'à épuiser le budget

Analogie : Au buffet, vous prenez les plats avec le meilleur "plaisir par euro" en premier.

Avantage : Super rapide

Problème : Peut rater la meilleure solution

[3] Méthode Dynamique (L'Intelligent)

Comment ça marche :

- Construit un grand tableau
- Compare intelligemment les choix

Analogie : C'est comme remplir un sudoku : chaque case aide à trouver les suivantes.

Avantage : Trouve LA meilleure solution ET reste rapide

Problème : Un peu plus complexe

Page 3 : La Force Brute en Détail

🔍 Comment ça Marche ?

La force brute teste **toutes les combinaisons possibles**.

Exemple avec 4 Actions (A, B, C, D)

Combinaisons à tester :

| | | |
|------------------|--------------------------|-----------|
| 1 action | : A, B, C, D | = 4 choix |
| 2 actions | : AB, AC, AD, BC, BD, CD | = 6 choix |
| 3 actions | : ABC, ABD, ACD, BCD | = 4 choix |
| 4 actions | : ABCD | = 1 choix |
| <hr/> | | |
| TOTAL = 15 choix | | |

⌚ Le Problème du Temps

Plus vous avez d'actions, plus ça prend du temps :

| Nombre d'actions | Combinaisons | Temps |
|------------------|---------------|---|
| 4 actions | 15 | Instantané <input checked="" type="checkbox"/> |
| 10 actions | 1 023 | Très rapide <input checked="" type="checkbox"/> |
| 20 actions | 1 048 576 | 5 secondes <input type="checkbox"/> |
| 30 actions | 1 073 741 824 | 1 heure <input type="checkbox"/> |

Conclusion : Impossible à utiliser avec beaucoup d'actions !

Page 4 : La Méthode Glouton en Détail

⚡ Comment ça Marche ?

La méthode glouton suit 3 étapes simples :

Étape 1 : Calculer la Rentabilité

Pour chaque action, on calcule :

$$\text{Rentabilité} = \text{Profit (\%)} \div \text{Prix}$$

Exemple :

Action-A : 30% de profit, coûte 60€
→ Rentabilité = $30 \div 60 = 0.50$

Action-B : 20% de profit, coûte 40€
→ Rentabilité = $20 \div 40 = 0.50$

Étape 2 : Trier

On trie les actions de la **plus rentable** à la **moins rentable**.

| | |
|-----------------------------|---------------|
| Action-5 : rentabilité 0.80 | ← Meilleure |
| Action-2 : rentabilité 0.60 | |
| Action-7 : rentabilité 0.40 | |
| Action-1 : rentabilité 0.20 | ← Moins bonne |

Étape 3 : Sélectionner

On prend les actions **une par une**, de haut en bas, tant qu'il reste du budget.

Budget de départ : 100€

1. Prendre Action-5 (40€) → Reste 60€
2. Prendre Action-2 (30€) → Reste 30€
3. Action-7 (50€) → Pas assez de budget
4. Prendre Action-1 (20€) → Reste 10€

Actions choisies : 5, 2, 1

Coût total : 90€

Page 5 : Exemple Complet - Méthode Glouton

Situation

Vous avez **100€** et ces 4 actions :

| Action | Prix | Profit % | Gain en € | Rentabilité |
|--------|------|----------|-----------|-------------|
| A | 40€ | 20% | 8€ | 0.50 |
| B | 50€ | 15% | 7.50€ | 0.30 |
| C | 30€ | 10% | 3€ | 0.33 |
| D | 60€ | 25% | 15€ | 0.42 |

Processus Étape par Étape

1. Trier par rentabilité :

```
A : 0.50 ← Meilleure
D : 0.42
C : 0.33
B : 0.30 ← Moins bonne
```

2. Sélectionner :

```
Prendre A (40€) → Reste 60€ ✓
Prendre D (60€) → Pas possible ✗
Prendre C (30€) → Reste 30€ ✓
Prendre B (50€) → Pas possible ✗
```

3. Résultat :

```
Actions achetées : A + C
Coût total : 70€
Gain total : 11€
Budget non utilisé : 30€
```

Page 6 : La Méthode Dynamique Expliquée

🎒 L'Analogie du Sac à Dos

Imaginez que vous partez en randonnée avec un **sac limité**.

Vous avez plusieurs objets :

- Chaque objet a un **poids** (= prix de l'action)
- Chaque objet a une **valeur** (= gain de l'action)

Question : Comment remplir votre sac pour avoir le maximum de valeur ?

📊 Comment ça Marche ?

On construit un **grand tableau** qui répond à toutes les questions :

"Avec les 5 premières actions et un budget de 200€,
quel est le meilleur gain possible ?"

Le tableau ressemble à ça :

| | Budget → 0€ | 50€ | 100€ | 150€ | 200€ |
|----------|-------------|-----|------|------|------|
| 0 action | 0 | 0 | 0 | 0 | 0 |
| Action 1 | 0 | (?) | (?) | (?) | (?) |
| Action 2 | 0 | (?) | (?) | (?) | (?) |
| Action 3 | 0 | (?) | (?) | (?) | (?) |

On remplit case par case en se demandant : "**Prendre ou ne pas prendre cette action ?**"

Page 7 : Exemple Complet - Méthode Dynamique

Situation Simple

3 actions et un budget de 100€ :

```
Action A : 20€, gain = 10€  
Action B : 40€, gain = 15€  
Action C : 60€, gain = 20€
```

Construction du Tableau

| | 0€ | 20€ | 40€ | 60€ | 80€ | 100€ |
|----------|--|-----|-----|-----|-----|------|
| 0 action | 0 | 0 | 0 | 0 | 0 | 0 |
| Action A | 0 | 10 | 10 | 10 | 10 | 10 |
| | (prendre A dès qu'on a 20€) | | | | | |
| Action B | 0 | 10 | 15 | 25 | 25 | 25 |
| | (à 40€: prendre B seule = 15) | | | | | |
| | (à 60€: prendre A+B = 25) | | | | | |
| Action C | 0 | 10 | 15 | 25 | 30 | 35 |
| | (à 100€: prendre B+C = 35) <input checked="" type="checkbox"/> | | | | | |

Meilleur résultat : 35€ de gain avec B + C !

Comment Retrouver les Actions ?

On remonte dans le tableau :

- 100€ : valeur = 35 → On a pris C (reste 40€)
- 40€ : valeur = 15 → On a pris B (reste 0€)
- 0€ : terminé !

Actions choisies : B + C

Page 8 : Comparaison des 3 Méthodes

Tableau Récapitulatif

| Critère | Force Brute | Glouton | Dynamique |
|---------------------|--|---|--|
| Vitesse |  Très lent | ⚡ ⚡ ⚡ Super rapide | ⚡ ⚡ Rapide |
| Solution | <input checked="" type="checkbox"/> Parfaite |  Bonne | <input checked="" type="checkbox"/> Parfaite |
| Facilité | <input checked="" type="checkbox"/> Simple | <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Très simple |  Complexe |
| 20 actions | 5 secondes | 0.0003 sec | 0.002 sec |
| 1000 actions |  Impossible | 0.002 sec | 0.02 sec |

Quelle Méthode Choisir ?

Pour Apprendre

→ **Force Brute** (facile à comprendre)

Pour la Rapidité

→ **Glouton** (ultra-rapide)

Pour la Perfection

→ **Dynamique** (meilleur des deux mondes)

Page 9 : Tests sur Vraies Données

Nous avons testé les algorithmes sur **2 fichiers de données historiques**.

📁 Dataset 1 : 1001 Actions

Qualité des données :

Total de lignes : 1001
✓ Bonnes données : 956 (95.5%)
✗ Données rejetées : 45 (4.5%)
(prix ou profit invalides)

Résultats :

| Qui ? | Actions | Coût | Gain |
|------------------|---------|------|-------------|
| Glouton | 13 | 499€ | 198€ |
| Dynamique | 13 | 499€ | 198€ |
| Sienna | 1 | 499€ | 197€ |

☒ **Notre algorithme gagne 1€ de plus que Sienna !**

📁 Dataset 2 : 1000 Actions

Qualité des données :

Total de lignes : 1000
✓ Bonnes données : 541 (54%)
✗ Données rejetées : 459 (46%)
(beaucoup de problèmes !)

Résultats :

| Qui ? | Actions | Coût | Gain |
|------------------|---------|------|-------------|
| Glouton | 18 | 500€ | 198€ |
| Dynamique | 18 | 500€ | 198€ |
| Sienna | 18 | 489€ | 194€ |

☒ **Notre algorithme gagne 4€ de plus que Sienna !**

Page 10 : Les Limites de Chaque Méthode

⚠ Force Brute : Trop Lent

Le problème : Avec 30 actions, il faut tester plus d'**1 milliard** de combinaisons !

| | |
|---------------------------|---|
| 10 actions → 1 seconde | ✓ |
| 20 actions → 5 secondes | ⚠ |
| 30 actions → 1 heure | ✗ |
| 1000 actions → Impossible | ✗ |

Conclusion : Ne marche qu'avec très peu d'actions.

⚠ Glouton : Pas Toujours Parfait

Le problème : Parfois, il rate la meilleure solution.

Exemple :

Budget : 100€

Action A : 60€, gain 18€, rentabilité 0.30
Action B : 50€, gain 10€, rentabilité 0.20
Action C : 50€, gain 10€, rentabilité 0.20

Glouton prend A → gain 18€
Meilleur choix = B+C → gain 20€ ✓

Différence : 2€ perdus

Mais : Sur nos vrais tests, le glouton a trouvé la même solution que le dynamique !

⚠ Dynamique : Utilise Beaucoup de Mémoire

Le problème : Crée un très grand tableau en mémoire.

| | |
|-----------------------------------|---|
| 20 actions → 10 KB de mémoire | ✓ |
| 1000 actions → 4 MB de mémoire | ✓ |
| 10000 actions → 400 MB de mémoire | ⚠ |

Conclusion : Parfait jusqu'à quelques milliers d'actions.

Page 11 : Gestion des Données Problématiques

Nettoyage Automatique

Notre programme rejette automatiquement les actions avec :

-  Prix = 0€ ou négatif
 -  Profit = 0% ou négatif
 -  Données manquantes
-

Résultats du Nettoyage

Dataset 1 :

```
Fichier : 1001 lignes
 Gardées : 956 (95.5%)
 Rejetées : 45 (4.5%)
```

Conclusion : Données de bonne qualité

Dataset 2 :

```
Fichier : 1000 lignes
 Gardées : 541 (54%)
 Rejetées : 459 (46%)
```

Conclusion : Beaucoup de problèmes, mais géré !

Avantage

Vous n'avez **rien à faire** ! Le programme nettoie tout automatiquement.

Page 12 : Nos Recommandations

⌚ Quelle Méthode Utiliser ?

Ça dépend de votre situation :

Situation 1 : Peu d'Actions (< 20)

Utilisez : Force Brute
Pourquoi ? C'est assez rapide et parfait

Situation 2 : Beaucoup d'Actions + Rapidité

Utilisez : Glouton
Pourquoi ? Ultra-rapide, résultat très bon

Situation 3 : Beaucoup d'Actions + Perfection

Utilisez : Dynamique
Pourquoi ? Rapide ET parfait

✉ Pour AlgolInvest&Trade

Notre conseil :

Clients Standards
└ Glouton (vitesse maximale)

Clients VIP / Gros Investissements
└ Dynamique (perfection garantie)

Tests / Vérifications
└ Force Brute (si < 20 actions)

Page 13 : Les Fichiers du Projet

📁 Programmes Python Crées

Nous avons créé **4 programmes** :

1. bruteforce.py

Force Brute
→ Teste toutes les combinaisons
→ Utiliser pour comprendre ou tester

2. optimized.py

Glouton
→ Super rapide
→ Utiliser en production

3. sac_a_dos_dynamique.py

Dynamique
→ Parfait ET rapide
→ Utiliser pour clients importants

4. analyse_datasets.py

Comparaison
→ Compare avec Sienna
→ Utiliser pour valider

Documentation

README.md

→ Guide complet

présentation.md

→ Présentation technique

pseudocode.md

→ Algorithme détaillé

Tous les codes sont :

- Bien commentés
 - Faciles à comprendre
 - Fonctions courtes (< 30 lignes)
-

Page 14 : Comment Utiliser les Programmes

🚀 Étape par Étape

Étape 1 : Préparer les Fichiers

Créer cette structure :

```
AlgoInvest&Trade/
├── bruteforce.py
├── optimized.py
├── sac_a_dos_dynamique.py
└── analyse_datasets.py
└── data/
    ├── actions.csv
    ├── dataset1_PythonP7.csv
    └── dataset2_PythonP7.csv
```

Étape 2 : Lancer un Programme

Ouvrir un terminal et taper :

Pour le glouton (rapide) :

```
python optimized.py
```

Pour le dynamique (parfait) :

```
python sac_a_dos_dynamique.py
```

Pour comparer avec Sienna :

```
python analyse_datasets.py
```

Étape 3 : Lire les Résultats

Le programme affiche :

```
⌚ Coût total : 498€
📈 Gain total : 197€
📊 Rentabilité : 39.5%  
📋 Actions à acheter :  
Action-5 : 45€, gain 16€  
Action-2 : 68€, gain 19€  
...

---


```

Page 15 : Résumé des Résultats

⌚ Ce Que Nous Avons Accompli

Objectifs Atteints

Vitesse

- └ Objectif : < 1 seconde
- └ Résultat : 0.002 secondes

Précision

- └ Objectif : Meilleur que Sienna
- └ Résultat : +3€ en moyenne

Robustesse

- └ Objectif : Gérer 1000+ actions
- └ Résultat : Jusqu'à 10000

📊 Chiffres Clés

- ⚡ 15 000x plus rapide que force brute
- 💰 +3€ de gain moyen vs Sienna
- ⌚ 100% des objectifs atteints
- Gère automatiquement les erreurs

⭐ Points Forts

1. **Rapide** : Réponse en moins d'1 seconde
 2. **Fiable** : Testé sur vraies données
 3. **Robuste** : Gère les données invalides
 4. **Simple** : Code facile à comprendre
-

Page 16 : La Notation Big-O Simplifiée

C'est Quoi Big-O ?

Big-O est une **façon de mesurer** la vitesse d'un algorithme.

Plus simple : C'est comme mesurer la vitesse d'une voiture.

Analogies Simples

O(1) - Constant

Comme ouvrir un placard
→ Toujours le même temps

O(n) - Linéaire

Comme lire un livre page par page
→ 2x plus de pages = 2x plus de temps

O(n log n) - Quasi-linéaire ← **Glouton**

Comme trier des cartes par valeur
→ Un peu plus lent que linéaire

O(n × k) - Polynomial ← **Dynamique**

Comme remplir un tableau case par case
→ Plus vous avez de lignes ET colonnes,
plus ça prend du temps

O(2^n) - Exponentiel ← **Force Brute**

Comme les poupées russes
→ Chaque poupée contient 2 autres poupées
→ Ça explose très vite !

Nos Algorithmes

| Algorithmme | Big-O | Explication Simple |
|-------------|-----------------------------|--------------------------------|
| Force Brute | $O(2^n)$ | Double à chaque action ajoutée |
| Glouton | $O(n \log n)$ | Trier puis parcourir |
| Dynamique | $O(n \times \text{budget})$ | Remplir un tableau |

Pourquoi C'est Important ?

Ça nous permet de **prédire** combien de temps prendra l'algorithme :

Force Brute avec 30 actions :

→ $O(2^{30}) = 1$ milliard d'opérations

→ Environ 1 heure

Glouton avec 1000 actions :

→ $O(1000 \times \log(1000)) = 10\ 000$ opérations

→ Environ 0.002 seconde

Page 17 : Le Pseudocode Simplifié

✍ C'est Quoi le Pseudocode ?

Le pseudocode, c'est comme une **recette de cuisine** pour l'ordinateur.

C'est entre le français et le code Python.

⌚ Algorithme Glouton en Pseudocode

DÉBUT

1. Lire toutes les actions du fichier
 2. Pour chaque action :
 Calculer rentabilité = profit % ÷ prix
 3. Trier les actions par rentabilité
(de la plus rentable à la moins rentable)
 4. Budget disponible = 500€
 Actions choisies = liste vide
 5. Pour chaque action (dans l'ordre) :
 SI prix de l'action ≤ budget disponible ALORS
 Ajouter l'action à "Actions choisies"
 Budget disponible = Budget disponible - prix
 FIN SI
 6. Afficher les actions choisies
- FIN

Algorithme Dynamique en Pseudocode

DÉBUT

1. Lire toutes les actions du fichier
2. Créer un grand tableau vide
(lignes = nombre d'actions)
(colonnes = budget possible de 0€ à 500€)
3. Pour chaque action :
 Pour chaque budget possible :

 Calculer 2 options :
 - Option A : Ne PAS prendre cette action
 - Option B : PRENDRE cette action

 Dans le tableau, mettre la MEILLEURE option
4. Remonter dans le tableau pour trouver
 quelles actions ont été choisies
5. Afficher les actions choisies

FIN

Pourquoi C'est Utile ?

Le pseudocode permet de :

- Comprendre la logique AVANT de coder
- Expliquer l'algorithme à quelqu'un
- Déetecter les erreurs facilement

C'est comme faire un **plan** avant de construire une maison !

Page 18 : Conclusion

Pour Résumer

Nous avons créé **3 algorithmes** pour trouver les meilleures actions :

- 1. Force Brute** : Parfait mais lent
 - 2. Glouton** : Rapide mais approximatif
 - 3. Dynamique** : Parfait ET rapide
-

Le Meilleur Choix

Pour **AlgolInvest&Trade**, nous recommandons :

Algorithme GLOUTON
pour la production

- ⚡ Ultra-rapide
- Résultats excellents
-  Géré données invalides

Avec l'option **Dynamique** pour les clients VIP qui veulent la perfection absolue.

Merci !

Questions ?

Tout le code et la documentation sont disponibles dans le [repository](#).

Bonne chance avec AlgolInvest&Trade ! 

Fin du Document