

Résolvez des problèmes en utilisant des algorithmes en python

Openclassrooms - Parcours Python - Projet n°7

Bérenger Ossété Gombé

10 août 2022

Sommaire

- 1 Introduction
- 2 Solution naïve
- 3 Solution optimisées
- 4 Comparaison de la solution
- 5 Conclusion

Présentation

Bérenger Ossété Gombé, 27 ans

- Baccalauréat Scientifique (2013)
- Maîtrise en informatique (2017)
- Spécialisation web chez Openclassrooms (janvier 2022)

AlgoInvest & Trade

Une société financière spécialisée dans l'investissement

- Objectif → optimiser ses investissements à l'aide d'algorithmes

Contraintes

- Une action ne peut être achetée plusieurs fois.
- Chaque opération d'achat est atomique.
- La société ne peut dépenser plus de 500€ par client.

Actions

Caractéristiques d'une action

Une action a est caractérisée par :

- Son nom $N(a)$ (exemple : Action-1)
- Son coût $C(a)$ (exemple : 20€)
- Son taux après deux ans $T(a)$ (exemple : 5%)

Bénéfices d'une action

Après deux ans, le bénéfice d'une action a est :

$$P(a) = C(a) \times T(a) \quad (1)$$

Investissements et bénéfice

Investissement

Un investissement est une liste d'actions à acheter à un client.

Bénéfice total

Soit un investissement de taille I tel que $|I| = N$ avec $N \geq 0$ Le bénéfice pour AlgoInvest & Trade est donc :

$$P(I) = \begin{cases} 0, & N \leq 0 \\ \sum_{k=0}^{N-1} cost_k \times rate_k, & \text{sinon} \end{cases} \quad (2)$$

et le coût est :

$$C(I) = \begin{cases} 0, & N \leq 0 \\ \sum_{k=0}^{N-1} cost_k, & \text{sinon} \end{cases} \quad (3)$$

Définition du problème

Trouver le meilleur investissement possible étant donné une liste d'actions.

Entrée

A une liste d'actions.

Sortie

Les actions $a_k \in I$ tel que $\sum_k P(a_k)$ soit maximale et $\sum_k C(a_k) \leq 500$.

Une première solution

Une solution *Bruteforce*

La solution la plus simple est d'énumérer toutes les possibilités puis de choisir la meilleure.

Principe

Générer **tout les investissements possibles** puis les trier et enfin choisir le meilleur.

Attention

Il faut générer les combinaisons et non pas les permutations.
Par exemple, ${}^5C_3 = 10$ et ${}^5P_3 = 60$.

Algorithme de la solution naïve

Pseudo-Code

```
1  BRUTEFORCE(actions , current , cost , profit , solutions , index)
2
3      si cost <= 500
4          solutions = solutions  $\cup$  {(current , cost , profit)}
5      fin si
6
7      pour i de index a |actions|
8          bruteforce(actions \ actions[i] ,
9              current  $\cup$  {action} ,
10                 cost + C(action) ,
11                 profit + P(action) ,
12                 solutions ,
13                 i)
14      fin pour
15
16      retourner solutions
17  FIN
```

Pour trouver le meilleur investissement :

```
1  MEILLEUR-CHOIX(actions)
2      solutions = BRUTEFORCE(actions ,  $\emptyset$  , 0 , 0 ,  $\emptyset$  , 0)
3      TRI(solutions)
4      retourner DERNIER-ELEMENT(solutions)
5  FIN
```

Complexité asymptotique en temps de la solution naïve

Forme générale

L'algorithme *bruteforce* (BF) est récursif :

$$BF(n) = |actions| \times BF(n-1) + \mathcal{O}(n) \quad (4)$$

Soit *TOTAL* le nombre d'appels récursifs de *bruteforce*

TOTAL =

$$|actions| \times (|actions| - 1) \times (|actions| - 2) \times \cdots \times 1 = |actions|!$$

Par conséquent, nous avons :

$$BRUTEFORCE = \mathcal{O}(n!) \quad (5)$$

Complexité asymptotique : une seconde approche

Nous énumérons toutes les combinaisons d'actions possibles.
Pour une entrée de $n = 20$ actions, nous avons $\binom{n}{k}$ solutions
avec k la taille de la sortie.

$$\binom{n}{k} = \frac{n!}{(n-k)! \times k!} \quad (6)$$

Complexité asymptotique en mémoire de la solution naïve

Dans le pire cas :

On ajoute au tableau de solutions une action à chaque itérations.

```
si cost <= 500
    solutions = solutions ∪ {...}
fin si
```

Taille finale du tableau

$$|actions| \times (|actions| - k) \times \dots \times 1 = |actions|!.$$

Conclusion :

$$BRUTEFORCE = \mathcal{O}(n!) \quad (7)$$

Conclusion de la solution naïve

Problèmes

- $\mathcal{O}(n!)$ grandit bien trop vite.
- On génère tout les cas, ce qui est efficace mais inefficent.

Conclusion

→ L'algorithme BRUTEFORCE n'est pas utilisable dans le monde réel.

Résolvez des problèmes en utilisant des algorithmes en python

Bérenger
Ossété Gombé

- Introduction
- Solution naïve
- Solution optimisées**
- Comparaison de la solution
- Conclusion

Résolvez des
problèmes en
utilisant des
algorithmes en
python

Bérenger
Ossété Gombé

- Introduction
- Solution naïve
- Solution optimisées
- Comparaison de la solution
- Conclusion

Résolvez des
problèmes en
utilisant des
algorithmes en
python

Bérenger
Ossété Gombé

- Introduction
- Solution naïve
- Solution optimisées
- Comparaison de la solution
- Conclusion