AlgoInvest&Trade

Sommaire

- Présentation du projet
- Algorithme de force brute
- Algorithme optimisé
- Comparaison backtesting

Présentation du projet

- Concevoir un algorithme qui maximisera le profit réalisé après deux ans d'investissement avec un budget maximum de 50 euros.

- Exigences client:

  Programmation en python

  Chaque action ne peut être achetée qu'une seule fois

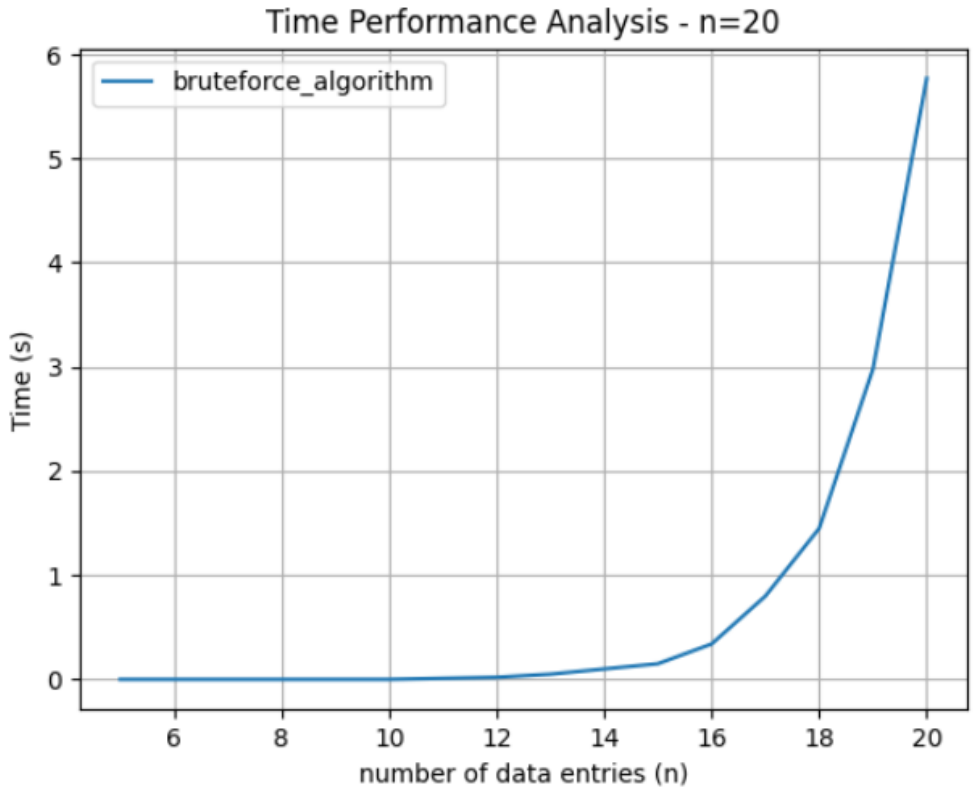
  Pas de possibilité de fractionner une action

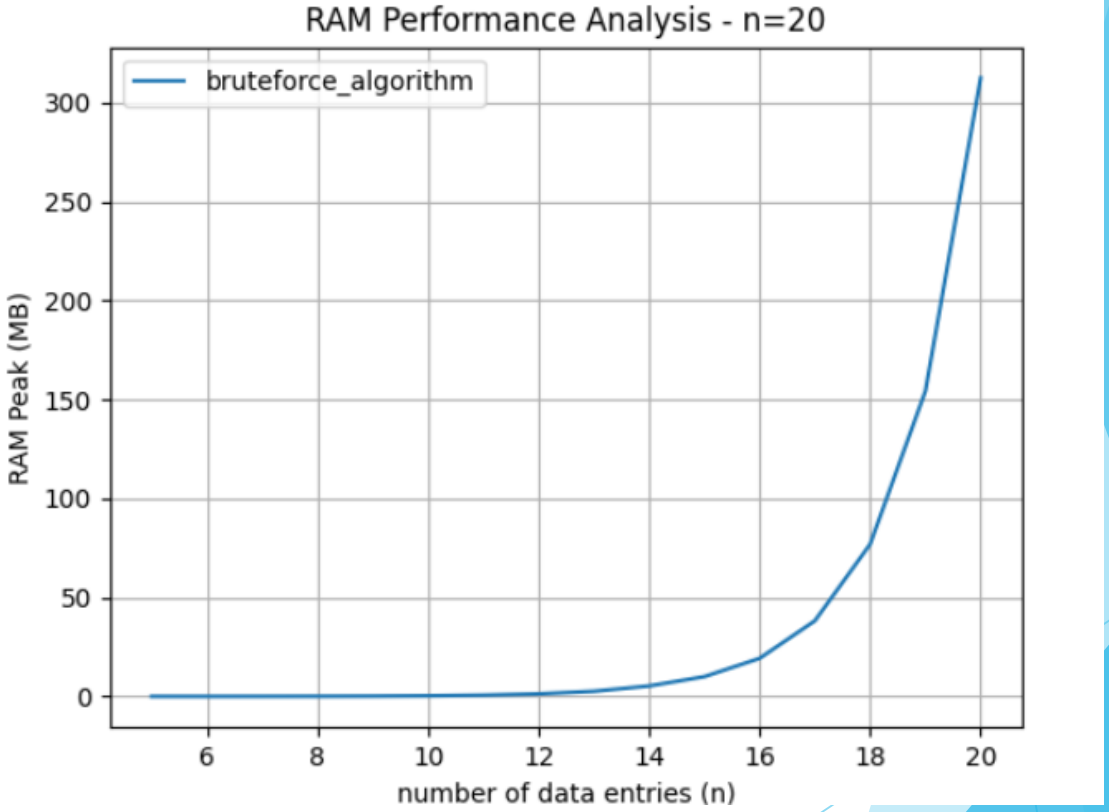
  Dépense maximum de 500 euros

Algorithme de force brute

- La fonction teste toutes les combinaisons possibles d'actions avec une boucle récursive. On calcul le coût total et le profit (coût * pourcentage)
- Elle ajoute ou supprime chaque action dans une liste temporaire (current_combinaison)
- 2^20 possibilités de combinaisons : 1 048 576
- Simple à comprendre, trouve la meilleure combinaison en testant tout.
- Très lent, inefficace si beaucoup d'actions

Algorithme de force brute





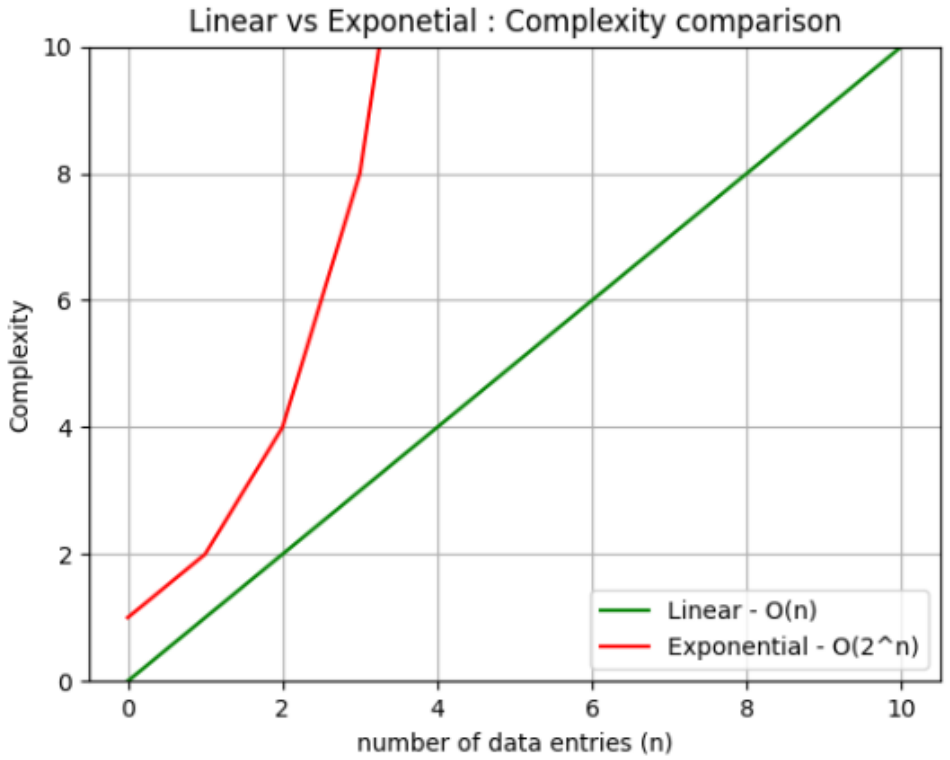
Algorithme optimisé

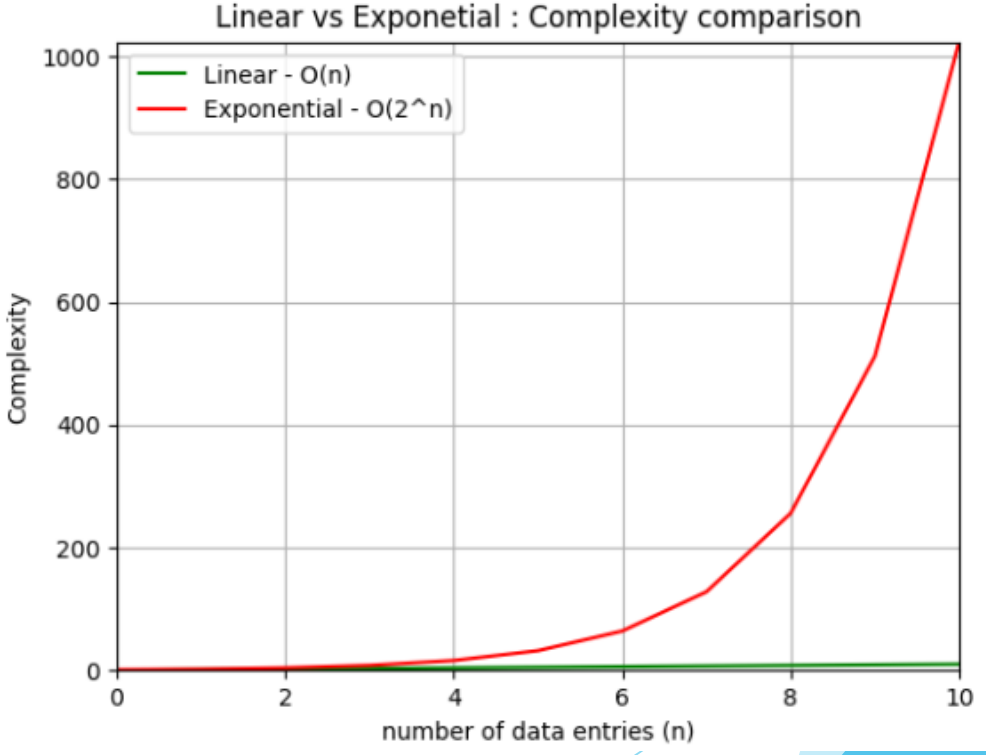
- Programmation dynamique
- Tableau 2D
- O(n)

Algorithme optimisé

```python
def knapsack_dp(max_budget, datas):
    """Utilise la programmation dynamique pour trouver la meilleure combinaison d'actions."""
    n = len(datas)
    max_budget = int(max_budget * 100)  # Convertir en centimes
    costs = [int(cost * 100) for _, cost, _ in datas]
    benefits = [int(cost * benefit * 100) for _, cost, benefit in datas]
    dp = [[0] * (max_budget + 1) for _ in range(n + 1)]
    keep = [[False] * (max_budget + 1) for _ in range(n + 1)]

    for i in range(1, n + 1):
        for w in range(max_budget + 1):
            if costs[i - 1] <= w and costs[i - 1] > 0:
                if dp[i - 1][w - costs[i - 1]] + benefits[i - 1] > dp[i - 1][w]:
                    dp[i][w] = dp[i - 1][w - costs[i - 1]] + benefits[i - 1]
                    keep[i][w] = True
                else:
                    dp[i][w] = dp[i - 1][w]
            else:
                dp[i][w] = dp[i - 1][w]

    w = max_budget
    selected_actions = []
    for i in range(n, 0, -1):
        if keep[i][w]:
            selected_actions.append(datas[i - 1])
            w -= costs[i - 1]

    total_benefit = dp[n][max_budget] / 100.0
    selected_actions.reverse()
    return {
        "total_benefit": total_benefit,
        "actions": selected_actions
    }
```

Analyse et performance





Comparaison : backtesting 1

optimisé

```
Share-FKJW - Coût: 21.08 €, Profit: 8.39 €
Share-GTQK - Coût: 15.40 €, Profit: 6.15 €
Share-USSR - Coût: 25.62 €, Profit: 10.14 €
Share-MTLR - Coût: 16.49 €, Profit: 6.59 €
Share-LPDM - Coût: 39.35 €, Profit: 15.63 €
Share-UEZB - Coût: 24.87 €, Profit: 9.81 €
Share-NHWA - Coût: 29.18 €, Profit: 11.60 €
Share-GHIZ - Coût: 28.00 €, Profit: 11.17 €
Share-KMTG - Coût: 23.21 €, Profit: 9.28 €

Total des coût: 499.95 €
Total des profits apres 2 ans: 198.44 €
```

SIENNA

- Action : share-GRUT
- Coût total : 498,76€
- Profit 196,61€

Comparaison : backtesting 2

optimisé

```
Share-ROOM - Coût: 15.06 €, Profit: 5.91 €
Share-XQII - Coût: 13.42 €, Profit: 5.30 €
Share-LXZU - Coût: 4.24 €, Profit: 1.68 €
Share-PLLK - Coût: 19.94 €, Profit: 7.96 €
Share-ZOFA - Coût: 25.32 €, Profit: 10.07 €
Share-FWBE - Coût: 18.31 €, Profit: 7.29 €
Share-IXCI - Coût: 26.32 €, Profit: 10.37 €
Share-ECAQ - Coût: 31.66 €, Profit: 12.50 €

Total des coût: 499.92 €
Total des profits apres 2 ans: 197.89 €
```

SIENNA

```
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€
```