# Projet 7

Bruteforce et Glouton avec Python

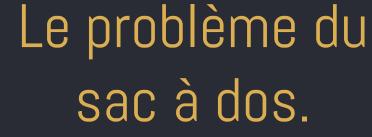
**LECOLE** Alexis

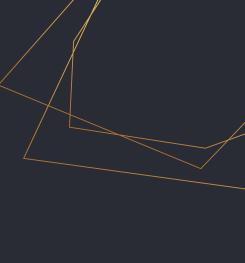
## SOMMAIRE

- Sac à Dos Présentation du problème du sac à dos
- 2. Bruteforce & Glouton Présentation du pseudo code.
- 3. Complexité algorithmique La notation big O.

- Algorithme
  Résolution des problèmes
  avec Python.
- 5 Résultats
  Rapport d'exploration des ensembles de données.
- 6. Conclusion









# 1/ Le problème du sac à dos.

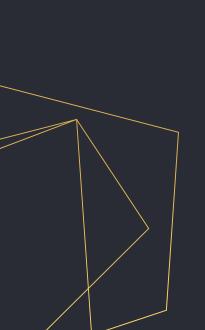
"En algorithmique, le **problème du sac à dos**, parfois noté (**KP**) (de l'anglais *Knapsack Problem*)1 est un problème d'**optimisation combinatoire**. Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou partie d'un ensemble donné d'objets ayant chacun un poids et une valeur.

Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum."

— WIKIPEDIA







Ouvre le CSV et stock le dans une variable sous la forme d'une liste de dictionnaire.

Lance le chrono pour mesurer le temps d'exécution de l'algorithme.

Récupère la taille de la liste obtenue.

Récupère un tableau qui stock le nombre de possibilité sous forme d'entier [0, 1, 2, ...]

Convertit chaque entier en binaire afin de garder une trace de tout ce qui a été tenté.

Ajoute les 0 manquant a chaque mot binaire (par exemple, si on ne choisit aucune action : 00000000000000000000) ainsi , toute les combinaisons sont référencées

Puis stock le dans un tableau "combinaisons"

----lere étape -----

Référencement de toute les combinaisons possibles =

Récupère le coût maximum. Initialise un tableau des combinaisons possible (vide pour l'instant).

pour chaque combinaison dans le tableau "combinaisons"
initialise le coût de la combinaison a 0
initialise le bénéfice de la combinaison a 0
pour chaque unité de la taille de la liste obtenue
si le chiffre obtenu est égale à 1
met à jour le coût de la combinaison
met à jour le bénéfice de la combinaison
si le coût de la combinaison n'excède pas le coût maximum
ajoute cette combinaison(coût, bénéfice)

-----2eme étape -----

recherche du meilleur investissement par algorithme de recherche du maximum =

initialise deux variables temporaire (coût, bénéfice) au premier élément du tableau combinaison possible pour chaque élément dans le tableau combinaison possible: si l'élément a un plus grand bénéfice que les variables temp met à jour les variables temp

-----3eme étape

affichage de la liste optimal des actions choisies =

initialise une liste des meilleurs investissements.

pour chaque itération dans la taille total du tableau des meilleurs investissements

si l'itération == chiffre l

ajoute à la liste des meilleurs investissements affiche la liste triée. met fin au chrono. affiche le chrono.

## 2.2/ Glouton: Pseudo-code

Ouvre le CSV.

Lance le chrono.

Stock le dans un tableau trie par ordre décroissant.

Initialise le coût total a 0. Initialise un tableau des meilleurs solutions.

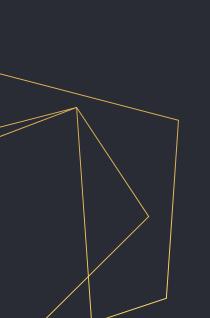
tant que le tableau a des occurrences et tant que le coût total n'excède pas 500

On stock l'occurence et son coût si le coût total + le coût de l'occurrence ne dépasse pas 500 on l'ajoute au tableau des meilleurs solutions on met à jour le coût total affiche la meilleur solution.

affiche le chrono.



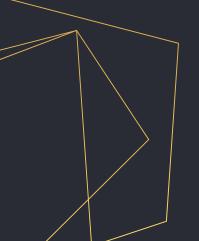




# 3/ Complexité Algorithmique.

La notion de complexité algorithmique se définit par les contraintes de temps et d'espace.





# 4/ Algorithme.

Bruteforce O(2<sup>(n)</sup>) complexité exponentielle

Glouton O(n log n) complexité linéarithmique

- → Temps: lent (environ 8 secondes)
- → Efficacité: toutes les possibilités sont testées
- → Résultats: optimal

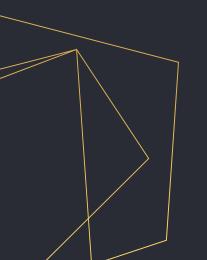
→ Temps: rapide (moins d'une seconde)

- → Efficacité: la plupart des possibilités sont testées
- → Résultats: satisfaisant

\*Données similaires (40 données) et même machine utilisée pour les deux algorithmes. → voir code.







Résultats.

## 5/ Résultats.

Résultat de Sienna:

Mon résultat:

Action:

Share-GRUT

Actions:

I'Share-MLGM'. 'Share-DBUJ'. 'Share-KGOI'. 'Share-OSPX'. 'Share-HITN'. 'Share-CBNY', 'Share-STKT', 'Share-KXOH', 'Share-FKJR', 'Share-RCCP', 'Share-VNON', 'Share-CYYC', 'Share-CIIX', 'Share-GVVO', 'Share-CUSU', 'Share-KFOG', 'Share-IYKD', 'Share-UEZE', 'Share-EVUW', 'Share-BMWW', 'Share-QQGZ', 'Share-NVDR', 'Share-IQMC', 'Share-HBXW', 'Share-YVIJ', 'Share-BXTP'. 'Share-HPOO'. 'Share-OLMP'. 'Share-PBXL'. 'Share-BGDY'. 'Share-FHZN', 'Share-JWVH', 'Share-AXIX', 'Share-HIGA', 'Share-UXBG', 'Share-OZBN', 'Share-ZNKU', 'Share-RNXT', 'Share-FYUX', 'Share-OOSO', 'Share-BXEV', 'Share-VVDF', 'Share-MKZB', 'Share-QDFN', 'Share-QGUP', 'Share-CGUS', 'Share-EMOV', 'Share-AIRL', 'Share-QSQG', 'Share-NDNV', 'Share-XJMO'. 'Share-AZTU'. 'Share-TXHO'. 'Share-EPCT'. 'Share-YHEL'. 'Share-KTGN', 'Share-GSGQ', 'Share-SIZB', 'Share-GIAJ', 'Share-HLYD', 'Share-IBVH', 'Share-QEDS', 'Share-VTWL', 'Share-HYQP', 'Share-CFOZ', 'Share-RUFN', 'Share-QPFG', 'Share-JKZM', 'Share-FTYF', 'Share-EIYB', 'Share-PSEN', 'Share-XZWG', 'Share-EMCQ', 'Share-OFCJ', 'Share-JPLO', 'Share-CSOA', 'Share-USUF', 'Share-BJIF', 'Share-LZLU', 'Share-ALZI', 'Share-VBED', 'Share-YNWM', 'Share-MGXU', 'Share-QOYO', 'Share-WEOQ', 'Share-RKNE', 'Share-JPLU', 'Share-QUQS', 'Share-CVLS', 'Share-TQVQ']

Coût total: 498.76 € Bénéfice total: 196.61 € Coût total : 499.93 € Bénéfice total: 2776.32 €

# 5/ Résultats.

#### Résultats de Sienna:

#### Mes résultats:

#### Actions:

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

Coût total: 489.24 €

Bénéfice total: 193.78 €

#### Actions:

['Share-LKSD', 'Share-DYVD', 'Share-JMLZ', 'Share-JWDZ', 'Share-ZLMC', 'Share-OCKK', 'Share-LXZU', 'Share-FUGM', 'Share-BBNF', 'Share-DSOO', 'Share-BMHD', 'Share-FFZA', 'Share-MEQV', 'Share-GEBJ', 'Share-SCWM', 'Share-FAKH', 'Share-KOVS', 'Share-DEPW', 'Share-KPBW', 'Share-CXYC', 'Share-VQQX', 'Share-BPPA', 'Share-GRVG', 'Share-DQXJ', 'Share-VVYP', 'Share-LAIC', 'Share-RBCS', 'Share-FCHD', 'Share-IWTG', 'Share-XQII', 'Share-FUDY', 'Share-LFXB', 'Share-ZGFP', 'Share-ROOM', 'Share-OWMP', 'Share-TGPO', 'Share-MZLD', 'Share-XYMR', 'Share-EOEN', 'Share-KRRA', 'Share-TQMM', 'Share-YIFQ', 'Share-FWBE', 'Share-OEYT', 'Share-JCWZ', 'Share-RWIW', 'Share-PILL', 'Share-BIJV', 'Share-GIXZ', 'Share-GYES', 'Share-PLLK', 'Share-TMRA', 'Share-ZKOZ', 'Share-DHIE']

Coût total: 499.88 €

Bénéfice total : 351.94 €

# Comparaison algorithmique avec Sienna

Il semble que l'algorithme de Sienna souffre davantage des erreurs dans le jeu de donnée, ou, il est possible que la solution de type glouton qu'elle propose ait un trop gros appétit.

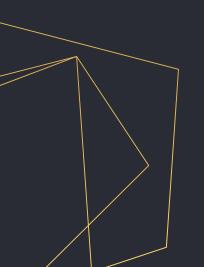
En effet, le fonctionnement du type glouton implique que si une action coûtant par exemple 498 euros rapporte 400 elle soit immédiatement choisi par l'algorithme.

Cependant, si 5 actions à 100 euros chacune rapportent 100 euros il est évidemment plus rentable de choisir ces dernières actions.

L'algorithme présenté ici évite d'avoir un trop gros appétit en calculant d'abord un pourcentage bénéfice/coût de l'action afin de déterminer les plus rentables en premier lieu.







Conclusion.

# Algorithme choisi

Nous choisirons un algorithme glouton dans notre cas, le jeu de donnée étant trop volumineux pour utiliser la force brute malgré un résultat optimal avec l'emploi de cette dernière.

L'algorithme glouton offre un résultat proche de l'algorithme de force brute en un temps nettement plus court.

Il convient donc de le choisir dans le cas présent.