Les algorithmes gloutons - Feuille d'exercices

▷ **Exercice 1** - Exercice débranché - ⋆

On suppose dans cet exercice, qu'on prend le système de pièces de monnaie européen (en centimes)

1. On suppose qu'on dispose d'autant d'unités que l'on veut. Quel est le rendu de monnaie proposé à l'aide d'un algorithme glouton de monnaie
   (a) si on doit rendre 2€47 ?
   (b) si on doit rendre 6€36 ?
   (c) si on doit rendre 3 €68 ?
2. On suppose à présent qu'on a un nombre limité de pièces réparti de la façon suivante :

Pièces en centimes	200	100	50	20	10	5	2	1
Quantité	3	5	9	8	6	5	4	1

   Quel est le rendu de monnaie proposé à l'aide d'un algorithme glouton de monnaie si on doit rendre successivement 2€47, 6€36 et 3€68 ?

▷ **Exercice 2** - Exercice débranché - ⋆

On considère un bureau de poste américain qui ne dispose pas de machine à affranchir mais des timbres de valeurs faciales :
1 cent, 10 cents, 21 cents, 34 cents, 70 cents et 100 cents
Un client veut affranchir un courrier pour 1$ et 40 cents.

1. Quel serait le rendu proposé à l'aide d'un l'algorithme glouton de rendu de monnaie ?
2. A-t-on obtenu une solution optimale ?

▷ **Exercice 3** - Exercice débranché - ⋆

« Ils sont fous ces Bretons ! »

Obélix dans Astérix chez les Bretons, Goscinny et Uderzo
(Source)

Nous sommes en 1966 après Jésus-Christ. Toutes les monnaies européennes sont décimalisées… Toutes ? Non ! Un royaume peuplé d'irréductibles Bretons résiste encore et toujours à la décimalisation de leur chère livre sterling. En effet, jusqu'en 1971, le Royaume-Uni ne possédait pas un système de monnaies décimalisées [1].
La livre(£) valait 20 shillings(s) et un shilling 12 pence(d). Il y avait d'autres sous-unités du penny et du shilling : voici le récapitulatif des "petits" billets et des pièces qui étaient utilisés en 1966.

	billets		Pièces							
Nom	a Fiver	a Quid	a Half-Crown	a Florin	a Bob	a Tanner	three Pence	a Copper	a half-penny	a farthing
Valeur	5 £	1 £	2 shillings and 6 pence	two Shillings	one Shilling	six Pence	$\frac{1}{4}$ shilling	one Penny	$\frac{1}{2}$ de penny	$\frac{1}{4}$ de penny
Valeur en pence										

1. Compléter les dernières lignes du tableau.
2. On se propose de tester un algorithme glouton de rendu de monnaie en prenant pour répartition de monnaie toutes les unités en pence décrites ci-dessus. On considère que l'on dispose d'autant d'unités que l'on veut.
   (a) On doit rendre 2 £ et 34 pences. Quel serait le rendu proposé par l'algorithme ?

1. Décimalisée signifie ici que la livre sterling est divisée en cent sous-unités : cent pence (au singulier : un penny). Un penny vaut un centième de livre.

(b) On doit rendre 3 £ et 4 shillings. Quel serait le rendu proposé par l'algorithme ?

(c) A-t-on obtenu des solutions optimales ?

▷ **Exercice 4** - Exercice branché - ⋆⋆

Une route comporte n stations-service, numérotées dans l'ordre du parcours, de 0 à $n-1$. Les distances entre chaque stations-service sont données par un tableau de données d $[d_0, d_1, ...; d_n]$ telle que :
la première est à une distance d_0 du départ, la deuxième est à une distance d_1 de la première, la troisième à une distance d_2 de la deuxième, etc. La fin de la route est à une distance d_n de la n-ième et dernière station-service.
Un automobiliste prend le départ de la route avec une voiture dont le réservoir d'essence est plein. Sa voiture est capable de parcourir une distance r avec un plein.

1. Donner une condition nécessaire et suffisante pour que l'automobiliste puisse effectuer le parcours. On la supposera réalisée par la suite.
2. En considérant 17 stations-service avec les distances d = [23, 40, 12, 44, 21, 9, 67, 32, 51, 30, 11, 55, 24, 64, 32, 57, 12, 80] et r = 100.
   L'automobiliste désire faire le plein le moins souvent possible. Écrire une fonction en Python nommée rapide de paramètre d et r qui détermine à quelles stations-service il doit s'arrêter.

▷ **Exercice 5** - Exercice débranché - ⋆

Le problème dit "du sac à dos" possède une importance majeure en algorithmique. Depuis plus d'un siècle, il fait l'objet de recherches actives et sa résolution trouve de nombreuses applications pratiques dans le domaine de la finance par exemple. Dans les années 1970, il fut a l'origine des premiers algorithmes de cryptographie à clé publique/privée. Nous tenterons de le résoudre par différentes "méthodes" gloutonnes.

Enoncé du problème

"Imaginons un voleur entrant de nuit dans un magasin ; il est équipé d'un sac pour réaliser son délit et y placer les objets volés. Malheureusement pour lui, son sac est usé et au-delà d'un certain poids, celui-ci risque de craquer. Dès lors, pour maximiser son profit, le voleur cherche à placer dans son sac les objets de plus grande valeur possible, tout en veillant à ne pas dépasser le poids maximal autorisé."

1. S'agit-il d'un problème d'optimisation ? Justifier la réponse.

Notations

On parle souvent du **KP** pour évoquer le problème du sac à dos **(KP = "Knapsack Problem")**.
On parle parfois du **0/1-KP** car les objets sont volés en entier (1) ou laissés dans le magasin (0) : le voleur ne peut pas prendre un bout d'objet ! Supposons que quatre objets(A, B, C, et D) présents dans le magasin, et que le voleur vole les quatre : la solution du **0/1-KP** peut-être notée {1, 1, 1, 1}

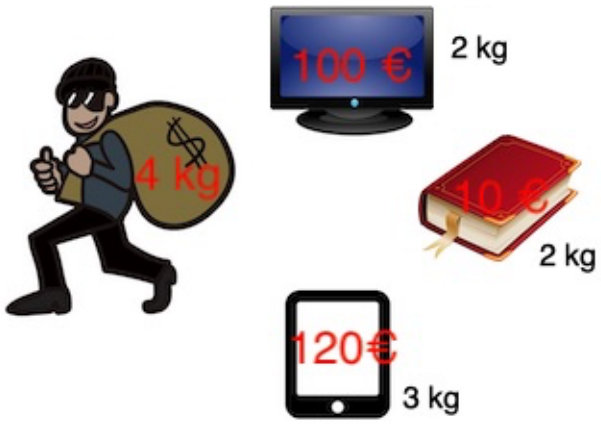
2. En utilisant les notations précédentes, quelle est la solution si le voleur vole les objets A et C ?
3. Soient trois objets de 4kg, 2kg et 1kg ainsi qu'un sac supportant jusqu'à 6kg. La solution {1, 0, 1} respecte-t-elle la condition sur la masse totale autorisée ?

▷ **Exercice 6** - Exercice débranché - ⋆

Prenons l'exemple d'un sac supportant 4 kg au maximum et un magasin avec trois objets A, B, C(Voir Figure 1). On supposera que le sac est suffisamment grand pour accueillir tous les objets. Pour la suite nous représenterons, les trois objets sous la forme d'un couple **(valeur en €, poids en kg) :**

Objet A : (100, 2) Objet B : (10,2) Objet C : (120,3).

FIGURE 1 –



1. En respectant le poids maximal, déterminer ("sur papier") le nombre de façons de remplir le sac (on pourra présenter les résultats dans un tableau).
2. Indiquer quel est le choix le plus profitable au voleur.

▷ **Exercice 7** Exercice débranché - ⋆ mais accès à la documentation officielle Python3

Soit la liste d'objets suivante où chaque objet est caractérisé par (un poids, une valeur) et un sac tel que $p_{max} = 2,7$:
L=[(0.2,300),(0.2,500),(0.1,250),(0.3,500),(1.3,2300),(0.8,2000),(1.3,2500)]

1. Ecrire une liste triée par valeurs (décroissante) puis une liste triée par rapport décroissant $\frac{\text{valeur}}{\text{poids}}$
2. Appliquer l'algorithme glouton sur ces deux listes et écrire une liste réponse uniquement constituée de 0 ou de 1. Commenter.
3. En Python, que renvoie les instructions `L.sort(key = lambda a : a[1])` et `L.sort(key = lambda a : a[0], reverse=True)` ?

▷ **Exercice 8** - exercice branché - ⋆⋆

Il s'agit d'exécuter un algorithme "naïf" qui consiste à tester toutes les choix d'objets acceptables pour finir par choisir la meilleure.

1. Télécharger sur l'ENT le fichier **ex8.py**.
2. Le code télechargé comporte plusieurs fonctions. La fonction **possibilites** de paramètre un entier n permet d'afficher le nombre de choix possibles de n objets (dans le sac du voleur).
   Tester la fonction **possibilites** pour $n \in \{3,4,5,6,7\}$. Conjecturer le nombre de façons de remplir le sac avec n objets.
3. La fonction **KPnaif** de paramètre une liste de couples **(valeur en €, poids en kg)** et un entier n retourne un triplet constitué de la solution optimale obtenue par l'algorithme naïf pour le sac, la valeur du sac et le poids du sac. Tester l'algorithme et noter les durées d'exécution sur les 5 premiers jeux de données du fichier datas.txt, c'est à dire pour $n \in \{4,7,8,10,15\}$.
4. A l'aide du logiciel de votre choix, représenter $t = f(n)$ où t représente la durée d'exécution du programme.
   Donner une équation d'une courbe de tendance de la forme $t = a \times e^{b \times n}$ (donner les valeurs des paramètres a, b).
5. Quel est le temps d'exécution prédit par ce modèle lorsque n =50? Commenter le résultat obtenu.