Devoir maison n°6

À rendre le vendredi 3 avril 2026

Exercice 1 *Un problème NP-complet : le problème du sac à dos. Les questions sont liées, mais les différentes parties peuvent êtres abordées indépendamment de la réussite des parties précédentes.*

On dispose d'un sac à dos d'un volume maximal donné, et d'un inventaire d'objets ayant chacun un certain volume et une certaine valeur. Le but de ce problème est d'étudier différentes stratégies permettant de choisir quels objets mettre dans notre sac afin de maximiser la valeur de son contenu. On est extrêmement fort à Tetris, de sorte que l'on considère que le volume occupé par plusieurs objets est la somme des volumes de chaque objet.

Pour traiter le problème en `Python`, on modélise les données de la manière suivante :

- La contenance du sac à dos est un entier `C` donnant son volume en litres,
- notre inventaire est modélisée par une liste de la forme

$$[\,[\text{"}\texttt{Objet1}\text{"}, \texttt{v1}, \texttt{m1}], [\text{"}\texttt{Objet2}\text{"}, \texttt{v2}, \texttt{m2}], ...]$$

où par exemple, `"Objet1"` est le nom du premier objet, `v1` est son volume (en litres) et `m1` sa valeur.

Partie A

1. Écrire une fonction d'entête `def Stats(Inv,Objet):` prenant en entrée l'inventaire `Inv` et le nom `Objet` d'un objet de l'inventaire, et renvoyant la liste `[v,m]` formée du volume `v` et de la valeur `m` de cet objet.
2. Écrire une fonction d'entête `def Verif(C,Inv,L):` prenant en entrée la contenance `C` du sac à dos, l'inventaire `Inv` considéré et une liste `L` de noms d'objets, et renvoyant `True` si le sac est assez volumineux pour contenir tous ces objets, et `False` sinon.
3. Écrire une fonction d'entête `def Depasse(C,Inv,L,Val):` prenant en entrée la contenance `C` du sac à dos, l'inventaire `Inv` considéré, une liste `L` de noms d'objets et un nombre `Val` et renvoyant `True` si le sac est assez volumineux pour contenir tous les objets de la liste `L` **et** si la somme des valeurs des objets de `L` dépasse `Val`, et `False` sinon.

Remarque culturelle : on peut coder une telle fonction `Depasse` *de sorte que son temps d'exécution soit assez rapide (tout ceci est quantifiable), on dit que le problème du sac à dos est de classe NP.*

Partie B

Pour choisir quels objets emporter, on décide d'étudier deux stratégies gloutonnes :

- Pour la stratégie S_1, on décide de prendre, à chaque étape, l'objet de plus grande valeur possible.
- Pour la stratégie S_2, on décide de prendre, à chaque étape, l'objet ayant le meilleur rapport valeur/volume.

4. Pour un sac de 22 litres, et l'inventaire :

$$[[\text{"}\texttt{Chocolat}\text{"}, 0.5, 900], [\text{"}\texttt{Cafe}\text{"}, 1, 90], [\text{"}\texttt{Serviette}\text{"}, 5, 100], [\text{"}\texttt{LivreMaths}\text{"}, 20, 700], [\text{"}\texttt{Casque}\text{"}, 4, 40]],$$

donner les noms des objets gardés dans le sac à dos ainsi que la valeur totale de son contenu, pour chaque stratégie ci-dessus.

5. Écrire une fonction d'entête `def Tri2(Inv):` prenant en entrée un inventaire `Inv` et renvoyant cet inventaire trié par valeur décroissante du rapport valeur/volume de ses objets.
6. Écrire une fonction d'entête `def Glouton2(C,Inv):` prenant en entrée la contenance `C` du sac à dos et un inventaire `Inv`, et renvoyant la liste `[Garde,Valeur]` formée :
   - de la liste `Garde` des noms des objets gardés en appliquant la stratégie gloutonne S_2,
   - de la somme `Valeur` des valeurs des objets gardés dans `Garde`.

Dans la suite de l'exercice, on considère qu'une fonction similaire `Glouton1(C,Inv)` *est codée appliquant la stratégie* S_1 *(mêmes entrées, même format de sortie).*

Partie C

On s'intéresse maintenant à la comparaison de ces stratégies pour des valeurs variables de la contenance `C` du sac à dos, et à inventaire fixé, mémorisé dans une variable globale `Inventaire`. Les fonctions `Glouton1` et `Glouton2` de la question 6 sont considérées correctement programmées.

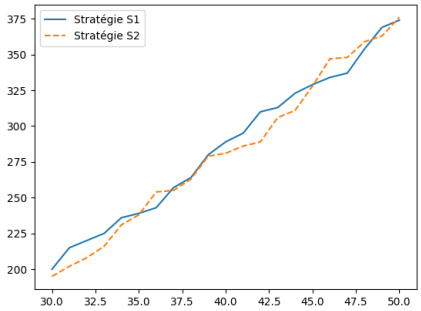
7. Écrire une fonction d'entête `def ListeValeurs(Cmin,Cmax):` prenant en entrée deux entiers `Cmin` et `Cmax` et renvoyant la liste `[Vals1,Vals2]` formée des deux listes `Vals1`,`Vals2`, où `Vals1` (resp. `Vals2`) est la liste

$$[\mathtt{V}_{\mathtt{Cmin}}, \mathtt{V}_{\mathtt{cmin+1}}, \dots, \mathtt{V}_{\mathtt{Cmax}}]$$

des valeurs totales $\mathtt{V}_{\mathtt{C}}$ des objets gardés par la stratégie S_1 (resp. S_2), pour la contenance `C` du sac à dos prenant les valeurs `Cmin`, `Cmin` + 1, ..., `Cmax`.

8. Écrire un programme permettant de tracer sur un même graphique les valeurs totales des objets gardés par chaque stratégie en fonction de la contenance `C` du sac à dos, pour `C` variant de 30 à 50.

9. Le programme précédent donne le graphique suivant. Qu'en déduire sur les stratégies S_1 et S_2?



Remarque culturelle : En partie A, on a vu qu'il était "assez simple" de vérifier la qualité d'une solution proposée. Par contre, le problème du sac à dos n'a pas de résolution optimale et algorithmiquement efficace connue. Si l'on en trouve une, ou si l'on démontre qu'il n'en existe pas, on résout l'un des problèmes les plus important des mathématiques du 20e siècle : le problème "P=NP".

Exercice 2 *Probabilités*

Une roue de loterie se compose de secteurs identiques numérotés de 1 à 12. Une personne fait tourner la roue devant un repère fixe. On suppose que chaque secteur a la même probabilité de s'arrêter devant ce repère.

À chaque partie, ce joueur mise une certaine somme d'argent en choisissant un, deux ou trois numéros sur les 12. Il est gagnant si le secteur s'arrêtant devant le repère porte l'un des numéros choisi. Ce joueur possède un crédit illimité, et effectue une suite de parties selon la stratégie suivante:

- Il mise sur le nombre 1 à la première partie.
- Pour tout $n \geq 1$, s'il perd à la n-ième partie, alors il mise sur les nombres 1 et 2 à la partie suivante. Sinon, il mise sur les nombres 1,3 et 5 à la partie suivante.

On admet l'existence d'un espace probabilisé $(\Omega, \mathcal{P}(\Omega), \mathbb{P})$ modélisant l'expérience aléatoire décrite ci-dessus.

1. On note, pour $n \geq 1$, p_n la probabilité de l'événement A_n : "le joueur gagne à la n-ième partie".

   (a) Montrer que p_n est non nul pour tout entier $n \geq 1$.

   (b) Soit $n \geq 1$. Déterminer $\mathbb{P}_{A_n}(A_{n+1})$ et $\mathbb{P}_{\bar{A_n}}(A_{n+1})$. En déduire :

$$\forall n \in \mathbb{N}^*, p_{n+1} = \frac{1}{12} p_n + \frac{1}{6}.$$

   (c) En déduire l'expression de p_n en fonction de $n \in \mathbb{N}^*$ et la limite de (p_n).

   Dans les questions suivantes, on fixe un entier $n \in \mathbb{N}^*$. On note, pour $k \in [\![1, n]\!]$, B_k l'événement " le joueur gagne une seule fois au cours des n premières parties et ce gain a lieu à k-ième partie."

2. Exprimer B_1 en fonction des événements A_k, pour $1 \leq k \leq n$.
3. En déduire $\mathbb{P}(B_1)$.
4. De même, calculer $\mathbb{P}(B_n)$.
5. Soit $k \in [\![2, n-1]\!]$. Montrer que $\mathbb{P}(B_k) = \dfrac{11}{96}\left(\dfrac{5}{6}\right)^{n-3}$.
6. En déduire la probabilité q_n pour que le joueur gagne exactement une seule fois au cours des n premières parties.

Exercice 3

Soit f la fonction définie sur $\mathbb{R}$ par $f(x) = \dfrac{e^x}{e^{2x}+1}$.

1. (a) Démontrer que f est paire sur $\mathbb{R}$.
   (b) Montrer que f est $\mathcal{C}^1$ sur $\mathbb{R}$ et étudier ses variations.
   (c) Justifier que l'équation $f(x) = x$ d'inconnue $x \in \mathbb{R}_+$ admet une unique solution $l \in \mathbb{R}_+$.
   (d) Montrer que $0 \leq l \leq \dfrac{1}{2}$. **Données :** $e^{\frac{1}{2}} \simeq 1,65$, et $e \simeq 2,72$ au centième.
2. (a) Montrer que $\forall x \geq 0, |f'(x)| \leq f(x)$.
   (b) En déduire $\forall x \geq 0, |f'(x)| \leq \dfrac{1}{2}$.
   (c) Montrer que $f([0, \dfrac{1}{2}]) \subset [0, \dfrac{1}{2}]$.

On définit la suite $(u_n)_{n\in\mathbb{N}}$ par

$$u_0 = 0 \text{ et } \forall n \in \mathbb{N}, u_{n+1} = f(u_n).$$

3. Montrer que pour tout $n \in \mathbb{N}$, $u_n \in [0, \dfrac{1}{2}]$
4. Montrer que $\forall n \in \mathbb{N}, |u_{n+1} - l| \leq \dfrac{1}{2}|u_n - l|$.
5. Montrer que $\forall n \in \mathbb{N}, |u_n - l| \leq \dfrac{1}{2^{n+1}}$.
6. En déduire la limite de (u_n).

— fin —