Les algorithmes gloutons
- EXERCICES –

Exercice 1 – Un algorithme possible du sac à dos

On reprend le problème du sac à dos étudié dans l'activité de découverte :

Un cambrioleur possède un sac à dos d'une contenance maximum de 30 kg. Au cours d'un de ses cambriolages, il a la possibilité de dérober 4 objets A, B, C et D. Voici un tableau qui résume les caractéristiques de ces objets :

Caractéristiques des objets :

objet	A	B	C	D
masse	13 kg	12 kg	8 kg	10 kg
valeur marchande	700 €	400 €	300 €	300 €

On souhaite déterminer les objets que le cambrioleur aura intérêt à dérober, sachant que :

- tous les objets dérobés devront tenir dans le sac à dos (30 kg maxi)
- le cambrioleur cherche à obtenir un gain maximum.

Une solution d'algorithme possible est d'utiliser l'algorithme glouton avec les hypothèses suivantes :

- 1 seul objet de chaque type est disponible (1 seul A, 1 seul B…);
- les valeurs de chaque objet sont stockées dans une liste triée dans l'ordre décroissant ($v_n > v_{n-1} > \dots > v_2 > v_1$).

Question 1 : Analyser l'algorithme suivant et son programme réalisé en Python :

Algo : ….

Prog : …

Exercice 2 – Le problème du rendu de monnaie

Le problème est le suivant :

Avec un jeu de n pièces de valeurs $p_n > p_{n-1} > \ldots > p_1$, on doit rendre un montant x en monnaie, quel est le plus petit nombre de pièces qu'il est nécessaire d'utiliser ?

C'est un autre problème d'optimisation dans lequel on doit **minimiser le nombre de pièces rendues**, il est différent du problème précédent du sac à dos où on devait maximiser le gain.

Une autre différence avec le problème du sac à dos, où un seul objet de chaque type était disponible, est qu'une pièce peut être utilisée plusieurs fois pour atteindre le montant à rendre.

Question 1 : On veut rendre **34** de monnaie avec le jeu de pièces suivant : [20, 10, 5, 2, 1], quelles pièces choisit-on ?

En minimisant le nombre de pièces, on choisit 1 pièce de 20, 1 pièce de 10 et 2 pièces de 2.

Question 2 : On veut rendre **17** de monnaie avec le même jeu de pièces que précédemment, quelles pièces choisit-on ?

On choisit 1 pièce de 10, 1 pièce de 5 et 1 pièce de 2.

Question 3 : On change le jeu de pièces [10, 6, 3], on veut rendre 17 de monnaie, quelles pièces choisit-on ?

On ne peut pas rendre 17 avec ce jeu de pièces. Il manque la pièce de valeur 1.

Question 4 : Donner une condition sur le jeu de pièces pour que tous les montants soient réalisables.

Il faut que la pièce 1 fasse partie du jeu de pièces disponibles pour pouvoir réaliser tous les montants possibles (avec l'hypothèse que le montant est toujours un entier).

Question 5 : A partir de l'exemple de la question 1, proposer une ou plusieurs méthodes (heuristiques) pour résoudre le problème de rendu de monnaie.

Méthode 1 : algo glouton vu dans le cours : choisir toujours les pièces de plus grande valeur possible.

De façon plus précise, s'approcher d'abord aussi près que possible du montant à rendre à l'aide de la pièce dont la valeur est la plus grande, puis utiliser ensuite les pièces de valeurs inférieures pour atteindre le montant.

Méthode 2 : énumérer toutes les combinaisons possibles, choisir celle qui utilise le moins de pièces (vu dans l'activité découverte)

Question 6 : Décrire par un algorithme glouton cette méthode pour résoudre le problème de rendu de monnaie (s'inspirer du programme fourni précédemment pour résoudre le problème du sac à dos).

Notation à utiliser :

- x : montant à rendre ;
- p : jeu de n pièces disponibles ;
- p_i : pièce i de la liste p (i de 1 à n);
- liste pièces rendues.

Données du problème :
En entrée : x et jeu de pièces
En sortie : liste des pièces rendues
Hypothèses :

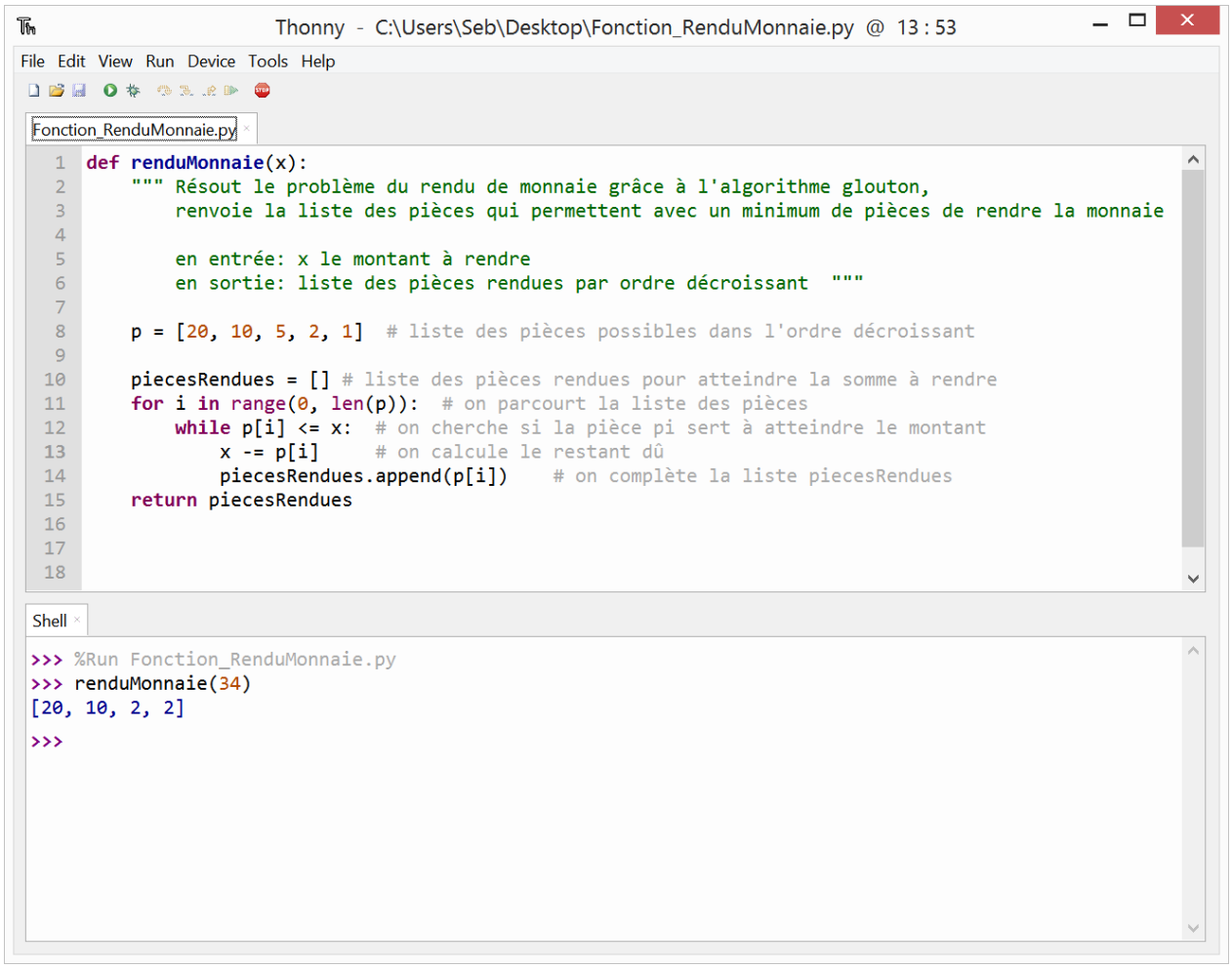
- le jeu de pièces est trié dans l'ordre décroissant $p_n > p_{n-1} > \ldots > p_1$;
- $p_1 = 1$.

Algorithme :

On veut donc s'approcher d'abord aussi près que possible du montant à rendre **x** à l'aide de la pièce $\mathbf{p_i}$ dont la valeur est la plus grande, au début c'est p_n, ensuite on va utiliser les pièces de valeurs inférieures pour atteindre le montant.

Pour l'ensemble des pièces
    Tant que p_i pièce la + grande <= x :
        x = x – p[i]
        stocker p_i dans liste pièces rendues

Question 7 : Utiliser la fonction fournie dans le programme Fonction_RenduMonnaie avec les différents jeux de pièces déjà utilisés :



Question 8 : Utiliser l'algorithme glouton avec le jeu de pièces suivant [37, 30, 21, 6, 4, 1], comment cet algorithme rend-il 51 en monnaie ? Quel serait le choix optimal ? Conclure ?

L'algorithme utilise 1 pièce de 37, 2 pièces de 6 et 2 pièces de 1 pour atteindre 51 (37 + 6+ 6 + 1 + 1 = 51) soit 5 pièces.

Il existe un nombre de pièces plus faible, le choix optimal est : 30 + 21

Donc l'algorithme glouton ne donne pas forcément le résultat optimal.

Ouvertures :

- on a simplifié le problème en partant d'une liste de pièces déjà triée, comment faire avec une **liste non triée** ?
- on peut utiliser des **préconditions** sur la liste des pièces dans la fonction RenduMonnaie.