Informatique : devoir n°7 (non surveillé)

Un **algorithme glouton** est un algorithme où, à chaque étape, on fait le choix local optimal.

Considérons par exemple le problème du voyageur de commerce qui doit visiter une liste de villes en essayant de parcourir le moins de distance possible. L'algorithme glouton associé consiste à visiter à chaque étape la ville non visitée la plus proche.

Un algorithme glouton n'est pas toujours optimal, mais il permet généralement d'obtenir une solution de bonne qualité en un temps raisonnable.

Exercice 1 - Problème du rendu de monnaie

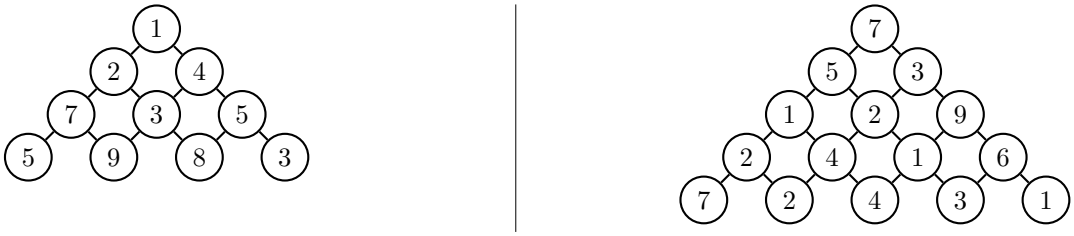
Les différentes valeurs des billets et des pièces en euros sont : 500, 200, 100, 50, 20, 10, 5, 2, 1. Le problème du rendu de monnaie est le suivant : on veut rendre un montant entier d'argent en utilisant le plus petit nombre possible de billets et de pièces.

Un algorithme glouton consiste à donner systématiquement le billet de plus grande valeur qui ne dépasse pas la somme restante. Par exemple, pour 29 euros, on commence par donner un billet de 20 euros, puis un billet de 5 et enfin deux pièces de 2.

1) Écrire une fonction `monnaie` qui, recevant un montant entier d'argent, renvoie une liste dont les éléments correspondent aux nombres de billets ou de pièces de chaque valeur rendus. Par exemple, `monnaie(29)` doit renvoyer la liste `[0, 0, 0, 0, 1, 0, 1, 2, 0]` (soit zéro billet de 500, 200, 100, 50, un billet de 20, zéro billet de 10, un billet de 5, deux pièces de 2 et zéro pièce de 1).

2) On peut démontrer (mais c'est assez difficile) que, lorsque la liste des pièces est $(1, 2, 5)$, cet algorithme glouton est optimal. Est-ce encore le cas lorsque la liste des pièces est $(1, 3, 4)$? On pourra considérer par exemple un montant à rendre de 6 euros.

Exercice 2 - La pyramide



On cherche à obtenir le total maximum dans une pyramide de nombres en partant du sommet et en descendant à chaque fois soit vers la gauche, soit vers la droite. On représentera une telle pyramide par une liste de listes : `[[1], [2, 4], [7, 3, 5], [5, 9, 8, 3]]` par exemple pour la première pyramide ci-dessus.

1) Algorithme glouton : on choisit à chaque étape le plus grand des deux nombres possibles. Par exemple, dans la première pyramide ci-dessus on obtient $1 + 4 + 5 + 8 = 18$ (ce n'est pas le résultat optimal qui est $1 + 2 + 7 + 9 = 19$).

a) Que donne cet algorithme glouton pour la deuxième pyramide ? Est-il optimal ?

b) Écrire une fonction `glouton` qui, recevant une pyramide représentée par une liste de listes, renvoie le total obtenu avec cet algorithme glouton.

```
>>> pyramide_1 = [[1], [2, 4], [7, 3, 5], [5, 9, 8, 3]]
>>> glouton(pyramide_1)
18
```

2) Programmation dynamique : pour trouver le total maximal on peut procéder comme suit. En partant de l'avant-dernière ligne on remplace chaque nombre par le total maximal obtenu en considérant les nombres du dessous. Ainsi pour la première pyramide la ligne `[7, 3, 5]` devient `[16, 12, 13]` ($7 + 9$, $3 + 9$ et $5 + 8$), puis la ligne `[2, 4]` devient `[18, 17]` ($2 + 16$ et $4 + 13$), et enfin la première ligne `[1]` devient `[19]` ($1 + 18$) qui nous donne le résultat maximal cherché.

a) Appliquer cette méthode à la deuxième pyramide.

b) Écrire une fonction `dynamique` qui, recevant une pyramide représentée par une liste de listes, renvoie le total maximal en utilisant cet algorithme.

```
>>> pyramide_1 = [[1], [2, 4], [7, 3, 5], [5, 9, 8, 3]]
>>> dynamique(pyramide_1)
19
```

3) Soit n le nombre de lignes de la pyramide. Déterminer la complexité asymptotique des fonctions `glouton` et `dynamique` en fonction de n.

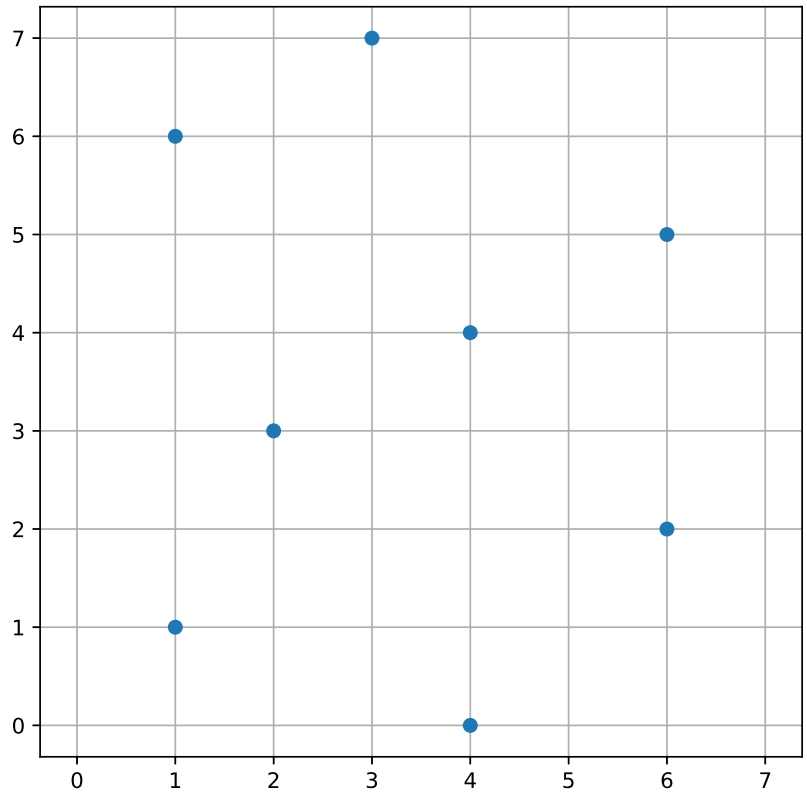
4) Sur le site de la classe on trouvera un fichier `pyramide.txt` qui contient une pyramide de hauteur 100. Quel est le total maximal que l'on peut obtenir en descendant cette pyramide ?

Exercice 3 - Le facteur de Manhattan

Les rues d'une ville sont agencées selon un réseau parfaitement orthonormal. Un facteur doit livrer des colis à différentes intersections en parcourant le moins de distance possible.

On modélise la ville par un plan muni d'un repère orthonormal. Les points de livraisons sont donnés sous forme de liste de points à coordonnées entières dont le premier élément est le point de départ.

La **distance de Manhattan** entre deux intersections est la longueur du plus petit chemin qui les relie en suivant les rues. Par exemple, la distance de Manhattan entre les points $(1, 1)$ et $(4, 0)$ est 4 (en allant par exemple de $(1, 1)$ à $(4, 1)$ puis de $(4, 1)$ à $(4, 0)$).



1) On suppose dans cette question que la liste des points de livraison est `[(1, 1), (4, 0), (6, 2), (2, 3), (4, 4), (6, 5), (1, 6), (3, 7)]`.

a) Dans quel ordre le facteur va-t-il livrer ses colis s'il choisit à chaque étape le point de livraison le plus proche (au sens de la distance de Manhattan) de l'endroit où il se trouve ?

b) Cet algorithme glouton est-il optimal ?

2) Écrire une fonction `distance_manhattan` qui, recevant deux couples d'entiers représentant deux points, renvoie la distance de Manhattan entre ces points. On rappelle qu'en Python la fonction valeur absolue est `abs`.

```
>>> distance_manhattan((1, 1), (4, 0))
4
```

3) Écrire une fonction `glouton` qui, recevant une liste de points de livraison, renvoie l'itinéraire suivant lequel le facteur parcourt ces points en choisissant à chaque étape le point le plus proche de l'endroit où il se trouve.

```
>>> glouton([(1, 1), (4, 0), (6, 2), (2, 3), (4, 4), (6, 5), (1, 6), (3, 7)])
[(1, 1), (2, 3), (4, 4), (6, 5), (6, 2), (4, 0), (3, 7), (1, 6)]
```