Algorithmes gloutons

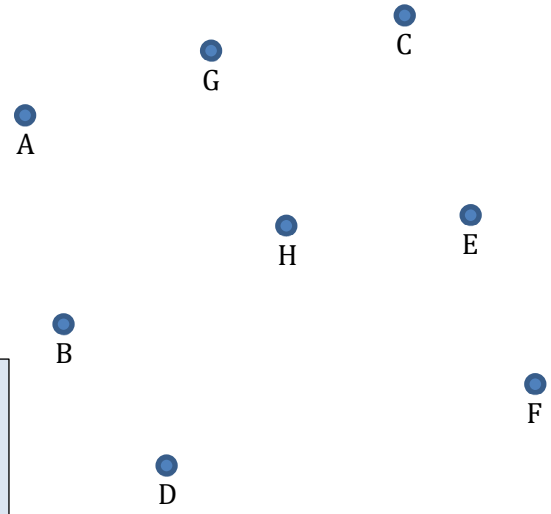
Nous avons vu dans l'activité sur le remplissage de caddie un exemple de problème d'optimisation : c'est-à-dire des problèmes pour lesquels l'algorithme doit permettre d'obtenir la « meilleure solution » parmi un grand nombre de solutions.
La qualité d'une solution peut être évaluée mathématiquement (distance totale parcourue, valeur du contenu d'un sac, nombre de pièces dans un rendu de monnaie….)

Il est possible de tester tous les choix possibles. Malheureusement les temps de calcul augmentent de façon vertigineuse avec la complexité du problème.

Nous pouvons alors utiliser la méthode des algorithmes gloutons dont le principe est, à chaque étape, de choisir l'option donnant le meilleur résultat partiel : on dit que l'algorithme fait un choix optimum local.

Problèmes classiques : le voyageur de commerce, le sac à dos, le rendu de monnaie.

Le voyageur : un voyageur doit passer par un ensemble de villes dont les distances entre deux villes sont toutes connues et revenir à son point de départ.



La « meilleure solution » est la plus courte : combien de chemins doit-on tester pour la trouver ?

La liste contient n villes, la première ville est fixée, c'est le point de départ du voyageur.

Combien de choix a-t-il pour commencer son voyage ?

On constate cependant que n'importe quel chemin existe en deux versions : dans un sens et dans l'autre. Il n'est pas utile de tester les deux versions. On peut donc _____.

Résultat : pour n villes, il faut tester possibilités.

Temps de calcul : Compléter le tableau avec le nombre de chemins à tester et le temps de calcul nécessaire en prenant 1 ms par chemin à tester.

n		Temps de calcul
5		
10		
15		
20		

L'algorithme glouton prendra dans ce cas la ville la plus proche à chaque étape (optimum local).
La solution ne sera pas optimale mais calculable, c'est ce qu'on appelle une « heuristique ».

Le sac à dos : le problème est celui de l'activité n°1 : le caddie. La méthode consiste à prendre à chaque étape le produit le plus cher au kilogramme.

Le glouton n'est pas optimal si on ne peut pas diviser les objets, il le devient si on peut prendre une portion d'un objet. (Voir l'activité n°1 et son corrigé)

Rendu de monnaie : il consiste à utiliser le moins de pièces et de billets possibles pour atteindre une certaine somme.
L'algorithme glouton va, à chaque étape, choisir la pièce ou le billet de plus grande valeur possible.
Dans presque tous les systèmes monétaires actuels, l'algorithme glouton donne la solution optimale.
Ils sont dits « canoniques ».

Exemple : rendre 9 euros.
Quelques possibilités sont :
9×**1** : 9 pièces
3×**1**+3×**2** : 6 pièces
2×**1**+1×**2**+1×**5** : 4 pièces
Etc ….

La solution optimale est : 2 pièces de **2** euros et 1 billet de **5** euros : 3 pièces.

L'algorithme glouton donne, en prenant à chaque étape la plus forte valeur disponible :
Etape 1 :
Etape 2 :
Etape 3 :

Système non canonique :
Avec un système basé sur les valeurs $[1,6,10]$, on veut rendre 12 euros.

- Que donne l'algorithme glouton ?

- Quel est le rendu optimal ?

Exercices :

1. Ecrire un programme en python qui permet de rendre la monnaie avec notre système monétaire. (Valeurs des pièces et billets en euros sans les centimes, de 1 à 200 euros). Une fonction devra prendre en paramètre la somme à rendre et renvoyer un tableau de tuples. Chaque tuple contiendra la valeur de la pièce ou du billet et le nombre correspondant. La fonction renverra également le nombre total de pièces ou de billets utilisés.
2. La valeur maximale d'un billet étant de 200 euros, utiliser le programme précédent pour déterminer le nombre maximum de pièces ou de billets à utiliser pour rendre une somme inférieure à 200 euros. La fonction devra renvoyer la somme à rendre et le nombre de billets ou de pièces à utiliser. Si plusieurs sommes donnent le même nombre de pièces/billets, la fonction devra toutes les fournir dans un tableau.
3. On a une liste d'activités qui peuvent se dérouler dans un gymnase. Deux activités sont compatibles si leurs créneaux horaires ne se recouvrent pas.
    a. Les créneaux sont 8h-13h ; 12h-17h ; 9h-11h ; 14h-16h ; 11h-12h. Combien peut-on placer d'activités au maximum dans une journée ?
    b. Simuler un algorithme glouton en choisissant à chaque étape, parmi celles compatibles, l'activité disponible qui se termine le plus tôt. Le résultat est-il optimal ?
    c. La liste de plages horaires est donnée sous la forme d'une liste de n tuple (début,fin). Ecrire un programme qui implémente l'algorithme précédent.
    d. Le tester avec la liste [(1,4), (0,6), (3,5), (12,13), (8,11), (8,12), (2,13), (6,10), (5,9), (3,8), (5,7), (13,16), (15,17), (16,19)]