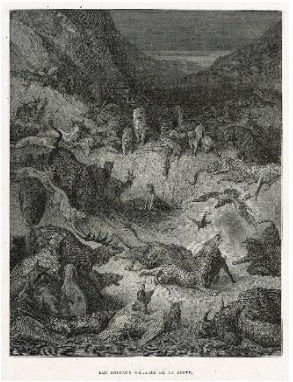
Les algorithmes gloutons

Au sens éthimologique du terme, le mot glouton désigne le gosier. Ce sens premier a dérivé sur le fait de manger sans retenu. Jean de La Fontaine dans Les animaux malades de la peste parle ainsi des appétits gloutons d'un lion qui dévorait des moutons et auquel il arrivait même de manger le berger. En faisant référence à cette gloutonnerie, nous sommes bien loin du numérique et pourtant …



Source : Site Wikipédia Domaine public
Gravure de Gustave Doré

Pourtant, l'adjectif glouton est attribué à des algorithmes très spécifiques qui ont la particularité d'agir « sans y aller avec le dos de la cuillère ». Une image - peu écologique- d'un algorithme glouton est celle de l'abattage d'un arbre qui peut se faire :

- Soit à la scie où l'on va finalement faire tomber l'arbre en produisant des grains macroscopiques (la sciure de bois) ;
- Soit à la hache « sans prendre de gants » en tombant l'arbre par gros copeaux successifs. En d'autres termes, à chaque coup de hache, on prend la part du lion.

Pour l'instant, rien n'indique de ce que l'on va faire de tout cela avec un ordinateur et pourtant …

Pourtant, nous allons donner un exemple très explicite d'algorithme glouton : Vous êtes hôtesse ou hôte de caisse dans un supermarché et pour un client qui doit payer 24 €, vous devez lui rendre la monnaie sur le billet de 100 € qu'il vient de vous remettre. Une première méthode consiste à « y aller à la petite cuillère » : lui rendre 6 pièces 1 €, 3 billets de 10 € et 2 billets de 20 €), une autre méthode consiste à « y aller à la grosse cuillère » : lui rendre 1 billet de 50€, 1 billet de 20€, 1 billet de 5 € et 1 pièce de 1€.

Nous avons donc mis en avant un algorithme qui consiste à procéder en plusieurs étapes, chaque étape consistant à remettre au client la plus grosse coupure ou pièce de façon à rester cependant dans le cadre de ce qu'on lui doit. Ainsi pour remettre la monnaie sur 100 € au client qui en doit 24 € :

1ère étape : nous lui rendons 50 € qui correspond à la plus grande coupure dont je dispose tel qu'elle soit inférieure ou égale à 76 €. Il nous reste à rendre au client la somme de 26 €.

2ième étape : nous lui rendons 20 € qui correspond à la plus grande coupure dont je dispose tel qu'elle soit inférieure ou égale à 26 €. Il nous reste à rendre au client la somme de 6 €.

3ième étape : nous lui rendons 5 € qui correspond à la plus grande coupure dont je dispose tel qu'elle soit inférieure ou égale à 6 €. Il nous reste à rendre au client la somme de 1 €.

4ième étape : nous lui rendons 1 € qui correspond à la plus grosse pièce dont je dispose tel qu'elle soit inférieure ou égale à 1 €. Nous n'avons plus rien à lui rendre.

Programmation de l'algorithme en Python

Nous vous proposons un développement de cet algorithme en Python. Pour ce faire, vous pouvez télécharger l'environnement Spyder accessible à l'adresse suivante : https://www.spyder-ide.org/ .

Nous restreignions volontairement la situation à rendre la monnaie sur des sommes dues qui sont des entiers.

Nous allons placer dans une liste nommée coupurespièces les différentes valeurs des coupures ou pièces dont nous disposons, ainsi la liste monnaie est la suivante :

coupurespieces = [500,200,100,50,20,10,5,2,1].

Notons P le montant en euros à payer par le client et V le montant en euros versée au départ par le client, montant sur lequel on doit lui rendre la monnaie. Nous allons construire une fonction nommée *arendre* qui donne la liste des billets et pièces de monnaie à rendre à un client qui doit payer P euros et qui a versé V euros. Cette liste est nommée *pourleclient.* Elle est vide au départ et va être alimentée successivement des valeurs correspondant à la plus grande valeur de la liste coupurespieces inférieure ou égale à ce qu'il reste à devoir au client.

De façon plus explicite, revenons à notre exemple pour lequel P = 24 et V = 100.
Au départ la liste *pourleclient* est vide.
On calcule V-P = 76. On recherche dans la liste *coupurespieces* le plus grand élément G qui est inférieure ou égal à R. Ici G = 50. On place cet élément dans la liste *pourleclient*.
On applique à présent la même procédure pour P=24 et V = V-G.
On continue jusqu'à ce que V soit égale à zéro.

Voici ci-dessous les lignes de codes de la fonction *arendre*. Ces lignes peuvent être copiées-collées dans la fenêtre d'édition du logiciel Spyder.

```
def arendre(P,V):
    # On définit la liste coupurespieces
    coupurespieces=[500,200,100,50,20,10,5,2,1]
    # Au départ la liste pourleclient est vide
    pourleclient=[]
    # Tant que (V-P) est différent de zéro on effectue la boucle
    while (V-P)!=0:
        i=0
        while coupurespieces[i]>(V-P) :
            i=i+1
        G=coupurespieces[i]
        pourleclient.append(G)
        V=V-G
    # La liste pourleclient est le résultat obtenu
    return(pourleclient)
```

Nous exécutons la fonction arendre appliquée au couple (24,100) et demandons au logiciel d'afficher la liste obtenue : print(arendre(24,120)) .

Conclusion

L'adjectif glouton est particulièrement bien choisi pour qualifier ce type d'algorithme. Il en va de même des mots choisis pour définir des objets du numérique qui sont la plupart très signifiants. Au même titre que d'autres disciplines, le numérique a ainsi son vocabulaire, vocabulaire qui fait partie de la culture numérique.

Jean-Alain Roddier
IA-IPR de mathématiques