Melcore

Zeste de savoir

Permuter deux variables sans en utiliser une troisième

16 novembre 2020

Table des matières

En informatique, il est une opération courante de vouloir permuter deux variables, et même si certain langage implémente cette fonctionnalité en leur sein (Python par exemple), on voit souvent les programmeurs et programmeuses recoder la permutation.

1. Permutation classique avec variables temporaires

Si nous devions imaginer une implémentation de la permutation, la première qui nous vient à l'esprit est surement celle-ci.

Imaginons que nous ayons 2 melons en mains, un melon vert dans la main droite et un melon jaune dans la main gauche, j'aimerais échanger de mains mes melons. Malheureusement les melons sont trop lourds et je ne peux qu'en mettre un dans chaque main. La meilleure solution, et vous en conviendrez que c'est celle que vous utiliseriez dans ce cas, consiste (1) à poser un des melons sur une table ce qui permet de libérer une main, (2) de changer de main le melon qui n'a pas été posé, et (3) de finalement reprendre le melon posé avec l'autre main. Ça se passe donc en 3 temps.

Prenons les variables a et b que nous voulons permuter.

Le pseudo-code suivant permet de permuter nos variables.

```
1 a = 5
2 b = 3
3 temp = a
4 a = b
5 b = temp
```

La variable temp est une variable dite temporaire, dans le sens où elle n'est utilisée que pour permettre la permutation, comme la table dans mon analogie fruitière. Elle permet de garder l'information comme on peut le voir dans le dérouler de l'exécution ci-dessous:

Ligne	a	b	temp
2	5	3	0

3	5	3	5
4	3	3	5
5	3	5	5

? Mais le titre de ce billet c'est "sans variable temporaire". Or là, on en utilise une.

Tous doux! Effectivement, et j'y viens, on peut permuter sans variable temporaire. En fait on peut le faire d'au moins deux façons en utilisant des propriétés mathématiques.

2. Permutation SANS variable temporaire

Bon on y est, deux façons donc!

On peut faire une permutation sans variable temporaire avec la propriété de deux opérations.

2.1. L'opération de l'addition (et soustraction)

La propriété intéressante de l'addition est la suivante.

$$a = a + b - bb = a + b - a$$

Bon avec seulement ça, ce n'est pas sûr que vous voyez où nous allons, alors je vais éclaircir un peu.

$$a = (a + b) - bb = (a + b) - a$$

$(a + b)$ peut nous servir de variable temporaire.

Soit l'algorithme suivant:

```
1 a = 3
2 b = 5
3 a = a + b
4 b = a - b
5 a = a - b
```

On remarque qu'il n'y a pas de variable temporaire. Mais la permutation est-elle bien effective? Pour s'en convaincre, écrivons l'exécution de l'algorithme avec les variables indicées de leur valeur au ième temps.

Ligne	**a**	**b**
2	a_0	b_0
3	$a_1 = a_0 + b_0$	$b_1 = b_0$
4	$a_2 = a_1$	$b_2 = a_1 - b_1 = a_0 + b_0 - b_0 = a_0$
5	$a_3 = a_2 - b_2 = a_1 - a_0 = a_0 + b_0 - a_0 = \mathbf{b}$	$b_3 = b_2 = \mathbf{a}$

À la fin, on a bien $a_3 = b_0$ et $b_3 = a_0$. La permutation est bien effective.

> Cette méthode peut poser problème selon la taille des objets a et b. Les nombres entiers étant bornés dans leur implémentation, cette borne peut etre dépassé à l'addition de a et b, ce qui causera un bug de l'algorithme.

2.2. L'opération XOR

2.2.1. keuwa ?

XOR est une opération binaire dite "bits à bits" appelés aussi le ou exclusif dont la valeur renvoie 1 quand les 2 bits sont différents. Sa table de vérité est:

p	**q**	**p XOR q**
0	0	0
0	1	1
1	0	1
1	1	0

Ainsi

$$1101 \ \ XOR \ \ 0101 = 1000$$

La propriété intéressante de XOR qu'on va utiliser est que:

$$p = (p \ \ XOR \ \ q) \ \ XOR \ \ q$$

ce qui peut se montrer facilement avec une table de vérité.

p	**q**	**p XOR q**	**(p XOR q) XOR q**
0	0	0	0
0	1	1	0

1	0	1	1
1	1	0	1

La première et la quatrième colonnes sont égales.

et aussi de manière plus anecdotique que

$$p \ \ XOR \ \ q = q \ \ XOR \ \ p$$

Soit l'algorithme suivant:

```
a = 0011
b = 1001
a = a XOR b
b = a XOR b
a = a XOR b
```

À la fin de cet algorithme, on a

$$b = (a \ \ XOR \ \ b) \ \ XOR \ \ b \Rightarrow b = a$$

et

$$a = (a \ \ XOR \ \ b) \ \ XOR \ \ ((a \ \ XOR \ \ b) \ \ XOR \ \ b) \Rightarrow a = (a \ \ XOR \ \ b) \ \ XOR \ \ a \Rightarrow a = b$$

On a bien la permutation qui a été effectué.

> *i* Cependant cela nécessite l'existence d'une fonction XOR pour les types utilisés.

La permutation est simple, mais parfois c'est dans les choses simples, qu'on se casse le plus la tête.

Merci pour votre lecture.