

Tony Hoare

Un bref aperçue
Par Théo.H

*⟨There are two methods in
software design. One is to make the pro-
gram so simple,
there are obviously no errors.
The other is to make it so complicated,
there are no obvious errors.⟩*



Édition scientifique avec L^AT_EX

21 février 2022

Table des matières

1	La logique de Hoare	1
1.1	Présentation	1
1.2	Méthode	2
1.3	Application	2
2	Le Quicksort	2
2.1	Presentation	2
2.2	Principe et implementation	3
2.3	Complexité	4

Une bref introduction

Tony Hoare ou *Charles Antony Richard Hoare* est un Informaticien influent du XX siecle. Il est principalement connue pour son implication sur la synchronisation des processus avec les moniteurs, La logique de Hoare pour des preuves formelles de justesse algorithmique et le *quicksort*.

1 La logique de Hoare

1.1 Présentation

Le but de la *Logique de Hoare* est de permettre un système formel de raisonnement sur la justesse d'un programme. Plus concrètement, comme chacun le sait, il existe des systèmes critiques pour lesquels l'incertitude n'est pas acceptable. Cette logique est basée sur les **triplet de Hoare**, ce triplet de la forme $\{P\}S\{Q\}$ est respectivement composé de :

- P Une precondition
- S Le programme
- Q Une postcondition

Les precondition et postcondition sont deux assertions appartenant à la **logique des prédicats**. Un triplet de Hoare est correct si la condition initiale P est vérifiée, exécuter S implique que Q est vrai.

exemple :

$$\{x = 5\}x := x \times 2\{x > 0\}$$

Ce triplet est clairement correct, en effet si $x = 5$ est que x est multiplié par deux, x est bien sûr supérieur à 0.

1.2 Méthode

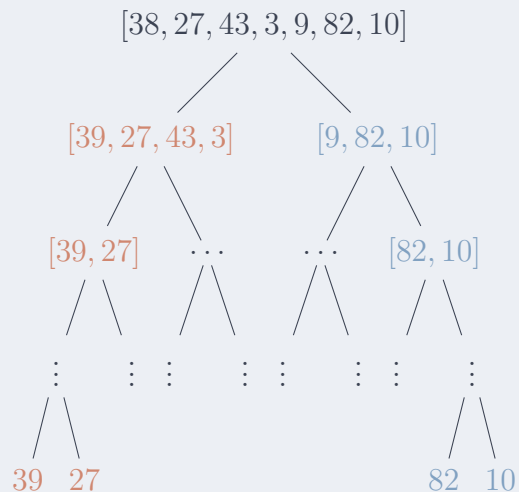
Maintenant ces triplet possede des propriete importante pour nous assurer de la bonne utilisation de ces lego dans le cadre de nos preuve de justesse.

1.3 Application

2 Le Quicksort

2.1 Presentation

Le *Quicksort* ou trie rapide dans la langue de Molière, est un algorithme de trie qui est ... rapide. De manière Plus concrète il s'agit d'un trie qui se rapproche des limites possible d'un trie comparatifs sans propriete connue.



Ce trie repose sur le principe du diviser pour mieux regner. Si en politique cela consiste a augmenter le nombre de candidats d'opposition, en informatique cela consiste en trois etapes :

1. **Diviser**, on divise l'instance en plus petite instance, en generale en separant en deux ce qui explique le retour recurent du \log_2 dans la complexiter de cette famille d'algorithme.
2. **Regner**,cette partie est surement la plus evidente car c'est la que l'ont le trie.
3. **Reunir**, Enfin reunie afin d'obtenir notre resultat final, c'est d'ailleur cette partie qui donne le caractere linaire des trie "*divide and conquere*"

Le trie dans l'exemple n'est pas le trie rapide, mais le trie fusion un trie baser lui aussi sur le *Divide and conquere*, il est a noter que la figure ne represente que la phase de division et pas la fusion elle meme, en effet celle ci est specifique au merge sort.

2.2 Principe et implementation

Le trie rapide, comme dit precedament, repose sur le *divide and conquere* plus particulierement autour d'un **pivot**. Ce pivot possede neanmoins une propriete interessante, a la suite d'un *partitionnement* tout les element a gauche lui sont inferieur et respectivment, ce que l'ont peut resumer par :

$$\forall (i, j) \in [p, q] \times [q + 1, r], \quad L[i] \leq L[j].$$

Cette propriete implique une infomation inportante, **tout les partition sont trier**, ainsi et de maniere naturelle. on peut juste ce contenter de reproduire cette procedure de maniere recursive.

```
int Partionner(liste t, uint p, uint r){
    int x=t.t[p];
    int i=p;
    int j=r;
    while (t.t[j]>x)
    {
        j--;
    }
    while (i<j)
    {
        swap(t.t, i, j);
        do
        {
            j--;
        } while (t.t[j]>x);
        do
        {
            i++;
        } while (t.t[i]<x);
    }
    return j;
}

void TriRapide(liste L, uint p, uint r){
    if (p<r)
    {
        q=Partionner(L,p,r);
        TriRapide(L,p,q);
        TriRapide(L,q+1,r);
    }
}
```

2.3 Complexité

De manière étonnante pour un trie utiliser aussi fréquemment sont pire cas est en $\Theta(n^2)$, néanmoins il existe de nombreuses variantes de ce trie qui résolvent ce problème. Un fix rapide est par exemple le choix du pivot de manière aléatoire.

Concernant la complexité dans le meilleur cas l'algorithme **partitionner** est en $\Theta(n)$, en effet les deux indices i et j sont respectivement incrémentés et décrémentés, et ceux jusqu'au pivot donc ces opérations se feront donc n fois. On peut s'étonner de ces résultats et se demander pourquoi cet algorithme est si utilisé.