Tony Hoare

Un bref apercue

‹There are two methods in software design. One is to make the program so simple,
there are obviously no errors.
The other is to make it so complicated,
there are no obvious errors.›

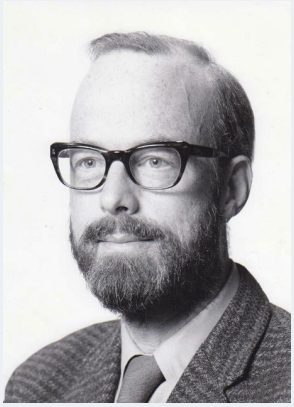


Table des matières

Une bref introduction

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

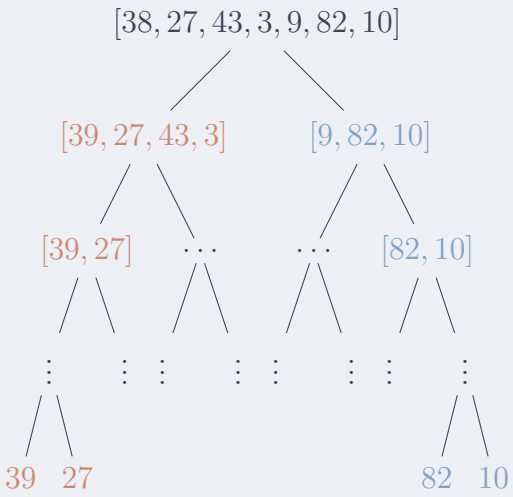
1 La logique de Hoare

test de la fesabilite

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'oposition, 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 charactere 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 reprensente 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 particulierment autour d'un **pivot**. Ce pivot possede neanmoins une

propriete interesante, a la suite d'un *partitionment* 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 manière 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 Complexite

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