

Le calcul des coûts (complexité des algorithmes)

1 Origine du mot

Le mot algorithme vient de Al-Kharizmi (IX^{ième} siècle) qui a inventé entre autre les chiffres arabes, les règles de trois, la racine carrée.

2 Analyse d'un algorithme

2.1 Par l'exemple, étudions un algorithme de tri par insertion.

```
Pour j allant de 2 à N Faire
  clé <- A[j]
  i <- j - 1

  Tant que i > 0 et A[i] > clé Faire
    A[i+1] <- A[i]
    i <- i - 1
  FinTantQue
  A[i+1] <- clé
FinPour
```

Estimer le temps d'exécution revient à estimer le temps d'exécution de chaque instruction et le nombre de fois que cette instruction est exécutée.

On distingue trois cas :

- Cas 1 : Le plus favorable ; la liste est déjà triée. Le coût de l'algorithme est alors linéaire par rapport à la taille des données.
- Cas 2 : Le plus défavorable ; la liste est triée à l'envers. L'algorithme est alors en $O(n^2)$.
- Cas 3 : Cas moyen. Ce cas se rapporte cas 2 en terme de complexité, $O(n^2)$.

Remarque importante : Les coûts c_i correspondent au coût réel de l'exécution de l'instruction i , mais ne change rien au fait que le coût moyen est de la forme quadratique. Les constantes c_i dépendent de la taille du tableau. C'est pour cela que l'on s'intéresse d'avantage à l'ordre de grandeur du temps d'exécution en fonction de la taille des données du problème.

2.2 Exemple de paradigme : Diviser pour mieux régner

Nous allons maintenant étudier le Tri-Fusion. L'approche "Diviser pour mieux régner" consiste à décomposer le problème initial en sous-problèmes de petite taille puis résoudre les problèmes décomposés un par un.

Nous avons dans cet algorithme, 3 étapes distinctes.

- 1^{ière} étape : diviser le problème en un certain nombre de problèmes.
- 2^{ième} étape : Régner sur les sous-problèmes où la résolution est plus triviale.
- 3^{ième} étape : Combiner les solutions des sous-problèmes pour obtenir une solution au problème initial.

```

Algorithme TriFusion(A,debut,fin)
  Si (debut < fin) Alors
    pivot <- (debut + fin) / 2
    TriFusion(A,debut, pivot)
    TriFusion(A,pivot+1,fin)
    Fusion(A,debut,pivot,fin)
  Finsi
  TriFusion(A,1,Taille(A))
Fin

```

Lorsqu'un algorithme contient un appel à lui-même (récursif), son temps d'exécution est souvent exprimé par une équation de récurrence.

Soit $T(n)$ le temps global d'exécution.

- Diviser : le coût est constant ($O(1)$).
- Régner : On résout successivement les 2 sous problèmes de taille $\lfloor \frac{n}{2} \rfloor$.
- Combiner : $O(n)$, coût linéaire.

3 Echelle de complexité

Nous donnerons ici les ordres de grandeur et leur correspondance :

- $O(1)$: Coût constant pour une instruction qui s'exécute une seule fois, ou bien en un nombre fini d'opérations.
exemple : L'accès à un élément dans un tableau, ou le calcul d'une expression.
- $O(\log(n))$: Le coût logarithmique quand on part d'un grand problème et qu'on le transforme en un sous-problème plus petit par réduction.
Exemples : Réduction dichotomique, ou opérations sur des arbres binaires de recherche.
Remarque : Même si N augmente, $\log(N)$ augmente très lentement.

- $O(n \cdot \log(n))$ Ce coût correspond aux algorithmes qui résument un problème en le divisant en sous-problèmes plus petits et en les résolvant indépendamment les uns des autres, et qui enfin combine les résultats.
Exemple : C'est le cas des tris récursifs.
- $O(n^2)$: Le coût est quadratique. Quand on traite les éléments par paire (2 boucles imbriquées par exemple).
Exemple : Manipulation de matrices, tri par comparaisons successives.
- $O(n^3)$: Le coût est cubique.
- $O(2^n)$: Coût exponentiel