

Chapitre 2 - Analyse d'algorithmes

Georges-Pierre BONNEAU (cours) - Mica MURPHY (note) - Antoine SAGET (note)

Lundi 15 Octobre 2018

Introduction

- **Prévoir les ressources nécessaires à l'exécution d'un algorithme : temps, mémoire, énergie, bande passante / réseau**
- **Hypothèse** : modèle de calcul générique, "machine à accès aléatoire" **RAM** ou **Random Access Memory**
 - Instructions arithmétiques en temps constant
 - Instructions de transfert de données en temps constant
 - Instructions de branchement conditionnel et inconditionnel (**goto**, **bal**) en temps constant
 - Les valeurs ont une taille limite maximum
 - On ne va pas tenir compte de la hiérarchisation de la mémoire

Analyse de complexité - Tri par insertion

Le temps d'exécution va changer en fonction de : - la taille du tableau - le contenu du tableau

Ligne	Temps constant par instruction	Nb d'exécutions
<i>L1</i>	C_1	1
<i>L2</i>	C_2	$N (j : 2 \dots N + 1)$
<i>L3</i>	C_3	$N - 1 (j : 2 \dots N)$
<i>L4</i>	C_4	$N - 1$
<i>L5</i>	C_5	$A = \sum_{j=2}^N t_j$
<i>L6</i>	C_6	$A - 1$
<i>L7</i>	C_7	$A - 1$
<i>L8</i>	C_8	$N - 1$
<i>L9</i>	C_9	$N - 1$

Avec t_j le nombre de fois où on rentre dans la sous boucle.

Soit $Temps(T)$ le temps de calcul total pour le tableau T $Temps(T) = C_1 + C_2 \times N + (C_3 + C_4 + C_8 + C_9) \times (N - 1) + C_5 \times A + (C_6 + C_7) \times (A - 1)$

Temps de calcul minimum (tableau déjà trié): - $t_j = 1$ - $A = N - 1$ - $Temps(N) = Constante \times N + Constante'$

Temps de calcul maximum (tableau déjà trié mais dans le sens inverse) : - $t_j = j$: comparaison de $Clef$ avec $T[j - 1 \dots 1]$ et à la fin on vérifie $i = 0 < 1$ - $A = \sum_{j=2}^N j = \frac{N(N+1)}{2} - 1$ - $Temps(N) = Constante \times N^2 + Constante' \times N + Constante''$

Analyse asymptotique. Comportement de $T(N)$ quand $N \rightarrow +\infty$

- **Pire temps :** $T(N) = O(N^2)$
- **Meilleur temps :** $T(N) = O(N)$

Notation O . Soient $f(N)$ et $g(N)$:

$$f(N) = O(g(N)) \Leftrightarrow \exists K > 0, \exists N_0 \in \mathbb{N}, \text{ tq } \forall N \geq N_0, f(N) \leq Kg(N)$$

Notation o . Soient $f(N)$ et $g(N)$:

$$f(N) = o(g(N)) \Leftrightarrow \forall K > 0, \exists N_0 \in \mathbb{N}, \text{ tq } \forall N \geq N_0, f(N) \leq Kg(N)$$

Exemples. $3N = O(\frac{N}{10})$ et $N = o(N^2)$

On ne s'intéressera toujours que au pire temps

Temps “moyen” du tri par insertion

$$t_j = \frac{j}{2}$$

Seuls les constantes changent :

$$\begin{aligned}\overline{T}(N) &= \overline{A} + \overline{B}N + \overline{C}N^2 \\ &= O(T(N)) \\ &= O(\overline{T}(N))\end{aligned}$$

- **Tri par insertion** (pire cas) : $T(N) = O(N^2)$
- **Tri par fusion** (pire cas) : $T(N) = O(N \ln(N))$

Illustration numérique. Tri d'un tableau de taille $N = 10^7$:

- Option A
 - 10^9 opérations par seconde (ordinateur “très rapide”)
 - tri par insertion
 - $T(N) = 2N^2$
 - Temps de calcul : $\frac{2 \times (10^7)^2}{10^9} \geq 55$ heures
- Option B
 - 10^7 opérations par seconde (100 fois plus lent que A)
 - tri par fusion
 - $T(N) = 50 \times N \times \ln(N)$
 - Temps de calcul : $\frac{50 \times 10^7 \times \ln(10^7)}{10^7} \approx 13$ minutes

Remarque. La complexité est donc primordiale !

Méthodes Diviser (récursivité) pour Régner

Remarque. Tri par insertion : algorithme incrémental

Soit $P(E)$ avec E un ensemble à N éléments et P un problème.

Méthode : 1) On découpe E en au moins 2 sous-ensembles A_1, \dots, A_k ($k \geq 2$) 2) On résoud $P(A_1), \dots, P(A_k)$ 3) On construit une sélection de $P(E)$ à partir des solutions de $P(A_1), \dots, P(A_k)$

Soit T le temps de calcul, $T = T(\text{division}) + \sum_{i=1}^k T(P(A_i)) + T(\boxed{(3)})$

La complexité de (1) et (3) doit être petite.

Tri-Fusion

Soit T un tableau de taille N

[Schéma d'un tableau coupé en deux]

- 1) **Division** : $N = \left\lfloor \frac{N}{2} \right\rfloor + \left\lceil \frac{N}{2} \right\rceil$
- 2) On trie les sous tableaux par appels récursifs
- 3) Fusion des 2 parties triées pour obtenir le tableau entier trié

Fusion : action (la donnée résultat A : un tableau de 1 à N d'entiers
les données p, q, r : 3 entiers)

```

{
  État initial :
    - 1 <= p <= q < r <= N
    - les valeurs Ap, ..., Aq sont triées
    - les valeurs Aq+1, ..., Ar sont triées
}
{
  État final :
    - les valeurs Ap, ..., Ar sont triées
    - les autres valeurs sont inchangés
}

```

Tri-Fusion : action (la donnée résultat A : un tableau 1 à N d'entiers
les données p, q, r : 2 entiers)

```

{ État initial : A contient A1, ..., AN, 1 <= p <= r <= N }

{
  État final :
    - A contient les mêmes valeurs (à une permutation près)
    - les valeurs Ap, ..., Ar sont triées
}

{ Lexique : q un entier }

{
  Algo :
    Si p = r : arrêt de la récursion
    Si p < r {
      q = floor((p + r) \ 2)
      Tri-Fusion(A, p, q)
      Tri-Fusion(A, q + 1, r)
      Fusion(A, p, q, r)
    }
}

{ Appel de la fonction : Tri-Fusion(A, 1, N) }

```

[Schéma du découpage du tableau {5,2,4,7,1,3,2,6} et appels en fonction des paquets]

Fusion(A, p, q, r)

```

{
  État initial :
    - 1 <= p < r <= N
    - Ap, ..., Aq trié
    - Aq+1, ..., Ar trié
}

{ État final : Ap, ..., Ar trié }

```

```

{
Lexique :
- G tableau d'entiers contenant  $A_p, \dots, A_q, +\infty$  ( $q-p+2$  éléments)
- D tableau d'entiers contenant  $A_{q+1}, \dots, A_r, +\infty$  ( $r-q+1$  éléments)
-  $i, j, k$  : 3 entiers
}

```

```

{
Algo :
Pour  $i$  allant de 1 à  $q - p + 1$  :
     $G[i] \leftarrow A[p + (i - 1)]$ 
 $G[q - p + 2] \leftarrow +\infty$ 

```

```

    Pour  $j$  allant de 1 à  $r - q$  :
         $D[j] \leftarrow A[q + j]$ 
    }
 $D[r - q + 1] \leftarrow +\infty$ 

```

```

 $i \leftarrow 1$ 
 $j \leftarrow 1$ 
Pour  $k$  allant de  $p$  à  $r$ :
    Si  $G[i] < D[j]$  :
         $A[k] \leftarrow G[i]$ 
         $i \leftarrow i + 1$ 

```

```

    Sinon :
         $A[k] \leftarrow D[j]$ 
         $j \leftarrow j + 1$ 

```

Exemple. Tableau $A = \{1, 3, 4, 8, 2, 5, 6, 9, 11\}$ - $G = \{1, 3, 4, 8, +\infty\}$ et $D = \{2, 5, 6, 9, 11, +\infty\}$ - k de 1 à 9 : $\{1, 2, 3, 4, 5, 6, 8, 9, 11\}$

```

1  $n \leftarrow 0$  ;
2  $u \leftarrow 3081, 45$  ;
3 tant que  $u < k$  faire
4   |  $u \leftarrow 1,04 \times u$  ;
5   |  $n \leftarrow n + 1$  ;
6 fin

```

.