Algorithmique

- Francesca Fiorenzi

Bureau I008B
Téléphone 01 69 33 61 18

http://www.lri.fr/~fiorenzi
fiorenzi@lri.fr

Objectifs

- Finaliser le package algorithmique de base d'un informaticien
- Retour sur certaines notions et approfondissements
- Structures et algorithmes

Programme

- Tris
- Récursivité
- Arbres binaires
- Adressage dispersé
- Automates

Organisation du cours

- *En TD* : Présentation en langage algorithmique et exemples
- *En TP* : Implémentation en Java
- *En projet(s)* : Expérimentations et exploitation des résultats

Références

- Polycopié du cours d’Algorithmique pour la première année de formation initiale
- Polycopié d’Eric Petitjean pour le module d'Algorithmique en deuxième année PEC de formation initiale

Cours 1 : Les algorithmes de tri basiques

Rappels

Situation de départ

- On dispose d’une structure conteneur, munie d'un itérateur (pour le parcours)
  - Liste, vecteurs, tableaux, ensembles, …
- Données contenues : ensemble muni d’un ordre total
  - Traduction Java : la classe implémente l’interface Comparable

Objectif

- Au départ :
  - Eléments dans un ordre quelconque

- A la fin :
  - Les **mêmes** éléments
  - Chaque élément est inférieur à celui qui le suit

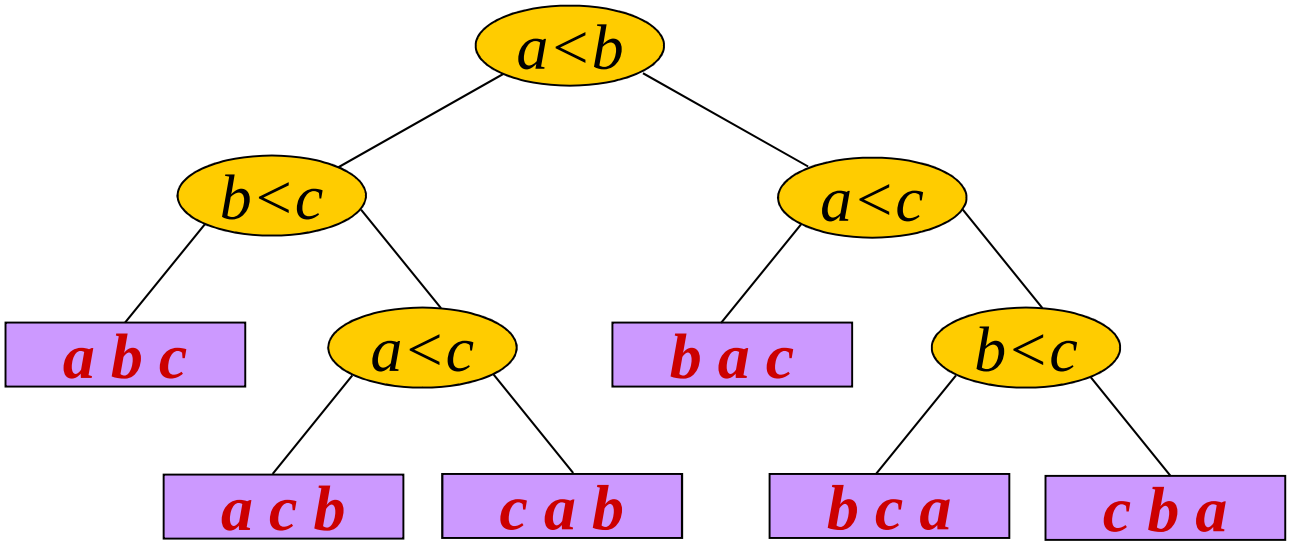
Idée du tri

- Réaliser des permutations (échanges) d'éléments pour passer de la situation de départ à la situation d'arrivée

- Objectif secondaire :
  - Le moins de permutations possible

Arbre (binaire) de décision

- Méthodes basées sur les comparaisons
- Arbre de comparaison



- Exécution d'un tri = parcours d'un chemin

Réflexions sur la complexité

- Il y a $n!$ façons d'arranger n éléments
- Un arbre binaire de hauteur h contient au plus 2^h feuilles
- Par conséquent, la profondeur d'un arbre contenant $n!$ feuilles est au minimum $\log_2(n!)$

Réflexions sur la complexité (2)

- Formule de Stirling : $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$
- Conclusion : $\log(n!) \sim n \log(n)$
- On ne peut pas faire mieux (en moyenne) pour la complexité d'un algorithme de tri que $n \log(n)$ où n est le nombre d'éléments à trier

Tris simples

Sélection

Insertion

Bulles

Le tri par sélection (dit “naïf”)

- *Principe :*
  - Pour chaque position successive dans le tableau, on cherche l’élément qui occupera cette position dans le tableau trié, et on l’y place en permutant cet élément avec l’élément courant
- *En version simple :*
  - On cherche le plus petit élément et on l’échange avec le premier
  - Puis on cherche le plus petit élément à partir du deuxième et on l’échange avec le deuxième
  - Etc…

Pourquoi ça marche ?

- Après le premier échange, le plus petit élément est en position 1
- Après le deuxième échange, les deux premiers éléments sont les deux premiers éléments du tableau trié
- Après le $i^{ème}$ échange, les i premiers éléments sont les i premiers éléments du tableau trié
- Arrivé à $i=n$, on a forcément fini (en fait même à $i=n-1$)

Un exemple

5	3	1	2	6	4
5	3	**1**	2	6	4
1	3	5	2	6	4
1	3	5	**2**	6	4
1	**2**	5	3	6	4
1	**2**	5	**3**	6	4
1	**2**	**3**	5	6	4

Un exemple (suite)

1	**2**	**3**	5	6	4
1	**2**	**3**	5	6	**4**
1	**2**	**3**	**4**	6	5
1	**2**	**3**	**4**	6	**5**
1	**2**	**3**	**4**	**5**	6

Rapide ou pas ?

- A l'étape *i* :
  - On cherche le plus petit élément entre les indices *i* et *n* : *n*-*i* lectures et comparaisons
  - Selon la position, on fait ou pas un échange
- Si on somme, on a au total :
  - Environ n^2 comparaisons (ne dépend pas du tableau)
  - Au pire *n* échanges, au mieux 0

Le tri par insertion

- *Principe :*
  - On va trier de façon itérative les *i* premiers éléments du tableau en mettant systématiquement le *i*ème élément à sa place parmi les *i*-1 premiers
- *En version simple :*
  - Evidemment le premier élément est trié
  - On trie les deux premiers éléments (simple)
  - On cherche la place que devrait avoir le troisième pour que les trois premiers éléments soient triés et on le déplace si besoin
  - Etc…

Pourquoi ça marche ?

- Après la première étape, les deux premiers éléments sont triés
- Après la deuxième étape, les trois premiers éléments sont triés
- ...
- Après la $i^{ème}$ étape, les i+1 premiers éléments sont triés
- Arrivés à $i = n-1$, le tableau entier est trié

Le même exemple

5	3	1	2	6	4
5	3	1	2	6	4
5	3	1	2	6	4
3	5	1	2	6	4
3	5	1	2	6	4
1	3	5	2	6	4
1	3	5	2	6	4

Le même exemple (suite)

1	3	5	2	6	4
1	2	3	5	6	4
1	2	3	5	6	4
1	2	3	5	6	4
1	2	3	5	6	4
1	2	3	4	5	6

Rapide ou pas ?

- A l'étape *i* :
  - On cherche la place du (*i*+1)-ème élément parmi les *i* premiers : au pire *i* comparaisons
  - On le met à sa place : au pire *i* décalages

- Au total :
  - Au pire $\Theta(n^2)$ comparaisons (au mieux *n*)
  - Au pire $\Theta(n^2)$ décalages (au mieux 0)

Le tri à bulles

- *Principe :*
  - On parcourt le tableau en arrangeant les éléments consécutifs, jusqu'à ce que plus aucun réarrangement ne soit nécessaire

- *En version simple :*
  - On parcourt tout le tableau en échangeant systématiquement les contenus des cases d'indice *k* et *k*+1 s'ils ne sont pas dans l'ordre
  - Tant qu'on a eu des changements à faire, on recommence

Pourquoi ça marche ?

- Après le premier passage, la plus grande valeur est à l'indice *n*
- Après le deuxième, les deux dernières cases sont les deux plus grandes valeurs du tableau, dans l'ordre
- ...
- Après la *i* ème étape, les *i* dernières valeurs du tableau sont les *i* dernières valeurs du tableau trié
- A *i* = *n*-1 au pire, on a fini
- Si aucun changement n'a été nécessaire, toutes les cases ont un contenu inférieur à celui de la case suivante
- C'est exactement la définition d'un tableau trié
- Donc le critère est bon aussi

Encore l’exemple

5	3	1	2	6	4
5	3	1	2	6	4
3	5	1	2	6	4
3	5	1	2	6	4
3	1	5	2	6	4
3	1	5	2	6	4
3	1	2	5	6	4

Encore l’exemple (2)

3	1	2	5	6	4
3	1	2	5	6	4
3	1	2	5	6	4
3	1	2	5	6	4
3	1	2	5	4	6
3	1	2	5	4	6
3	1	2	5	4	6

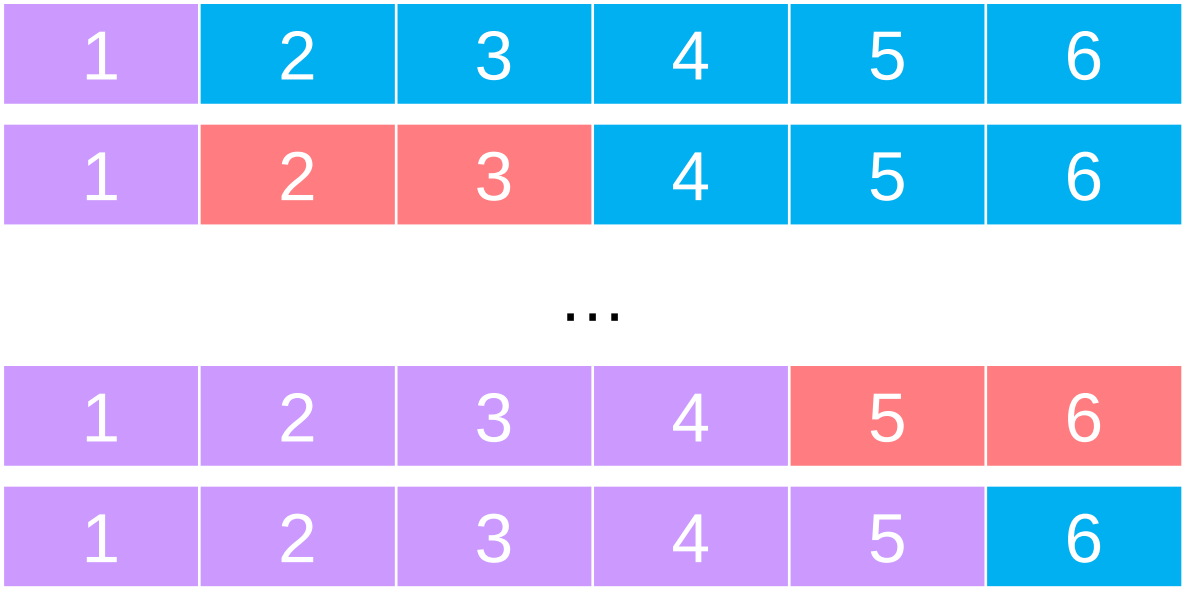
Encore l'exemple (3)

3	1	2	5	4	6
1	3	2	5	4	6
1	3	2	5	4	6
1	2	3	5	4	6
1	2	3	5	4	6
1	2	3	5	4	6
1	2	3	5	4	6

Encore l’exemple (4)

1	2	3	5	4	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6

Encore l’exemple (5)



Fin : aucun réarrangement pour ce dernier parcours

Rapide ou pas ?

- A l'étape *i* :
  - *n*-1 comparaisons (indépendant du tableau)
  - Au pire *n*-*i* échanges (au mieux 0)

- Au total :
  - $\Theta(n^2)$ comparaisons
  - Au pire $\Theta(n^2)$ échanges (au mieux 0)

Variantes

- *Optimisation facile :*
  - A l'étape *i*, on ne parcourt que de l'indice 1 à l'indice *n*-*i*+1 puisque le reste est déjà à sa place (le gain de temps est non négligeable)
- *Tri bulles descendant :*
  - On prend les indices dans l'ordre inverse (équivalent)

Variantes (2)

- *Tri « Shaker » ou « Shuttle » :*
  - On reprend au début dès qu'un échange a été détecté
- *Tri bidirectionnel ou « Boustrophedon » :*
  - Une fois en montant, une fois en descendant, etc..
- Seule la première variante fait gagner du temps