

Classement des algorithmes de tri

Site Internet :
www.gecif.net

Type de document :
Exercice

Intercalaire :

Date :

I – Comparaison des 3 algorithmes de tri classiques

Les 3 algorithmes de tri classiques sont :

- le tri par **sélection**
- le tri par **insertion**
- le tri à **bulles**

Dans le meilleur des cas (liste déjà triée) et dans le pire des cas (liste rangée à l'envers) chacun des ces 3 algorithmes de tri effectue un nombre précis de comparaisons et un nombre précis d'échanges en fonction du nombre total de valeurs à trier n.

Les 3 premiers tableaux donnent le nombre de comparaisons et le nombre d'échanges pour n=6, n=10 et n=12 : à tester réellement avec les cartes en papier, et à compter manuellement en effectuant tous les mouvements réalisant chacun des 3 algorithmes.

Pour n=6	Dans le meilleur des cas		Dans le pire des cas	
	comparaisons	échanges	comparaisons	échanges
Tri par sélection				
Tri par insertion				
Tri à bulles				

Pour n=10	Dans le meilleur des cas		Dans le pire des cas	
	comparaisons	échanges	comparaisons	échanges
Tri par sélection				
Tri par insertion				
Tri à bulles				

Pour n=12	Dans le meilleur des cas		Dans le pire des cas	
	comparaisons	échanges	comparaisons	échanges
Tri par sélection				
Tri par insertion				
Tri à bulles				

Ce dernier tableau récapitule les valeurs en fonction de n dans le cas général : quelque soit la valeur de n.

Pour n quelconque	Dans le meilleur des cas		Dans le pire des cas	
	comparaisons	échanges	comparaisons	échanges
Tri par sélection				
Tri par insertion				
Tri à bulles				

II – Complexité temporelle des 13 algorithmes de tri

Les 13 algorithmes de tri comparés ici sont :

- 1 : le tri par Sélection
- 3 : le tri par Insertion et ses 2 variantes le tri Shell et le tri Gnome
- 4 : le tri à Bulles et ses 3 variantes le tri Shaker (tri à bulles bidirectionnel), le tri À peigne (tri à bulles à pas variable) et le tri Oyelami (tri à peigne bidirectionnel)
- 1 : le tri Fusion (basé sur le principe "diviser pour mieux régner")
- 1 : le tri Rapide (utilise un pivot) : place à gauche du pivot toutes les valeurs inférieures et à droite les valeurs supérieures
- 2 : les tris utilisant un arbre binaire : le tri Maximier (rempli un arbre binaire dont le parcours en largeur donne les noeuds dans l'ordre) et l'extraction du parcours en profondeur dans l'ordre infixe d'un Arbre Binaire de Recherche (ABR+infixe)
- 1 : le tri Timsort (utilisé par Python : fonction sorted() et méthode sort() des listes) : tri fusion suivi d'un tri par insertion

Le tableau suivant récapitule la complexité temporelle des 13 algorithmes de tri en distinguant les 3 cas suivants :

- dans le meilleur des cas (liste déjà triée dans l'ordre croissant au début)
- en moyenne (après un grand nombre de tests avec des listes mélangées)
- dans le pire des cas (liste triée dans l'ordre décroissant au début)

<i>Complexité temporelle</i>	Meilleur des cas	En moyenne	Pire des cas
Sélection			
Insertion			
Bulles			
Shaker			
À peigne			
Gnome			
Shell			
Oyelami			
Maximier			
ABR + infixe			
Fusion			
Rapide			
Timsort			

Rappel des différents types de complexités temporelles :

- logarithmique : **$O(\log(n))$**
- linéaire : **$O(n)$**
- quasi-linéaire : **$O(n \cdot \log(n))$**
- quadratique : **$O(n^2)$**

Prise de notes complémentaires et personnelles :