

Dernière mise à jour	Informatique	Denis DEFAUCHY – Site web
30/03/2023	8 - Tris	Résumé

Informatique

8

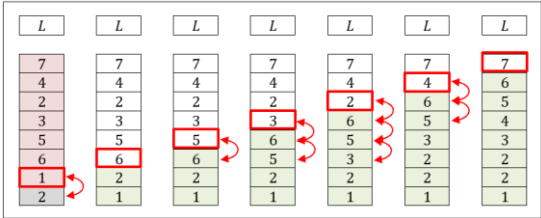
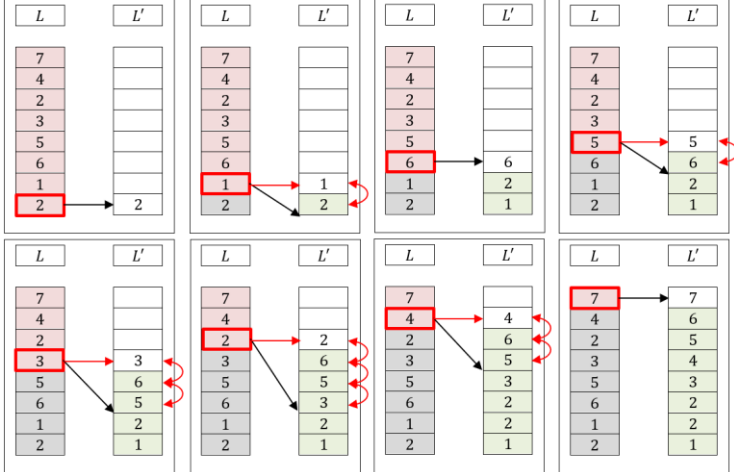
Tris

Résumé

Dernière mise à jour	Informatique	Denis DEFAUCHY – Site web
30/03/2023	8 - Tris	Résumé

Définitions	
Clef	C'est ce qui est utilisé pour trier des éléments
Tri comparatif	Tri fondé sur la comparaison entre les « clefs » des éléments pour les trier
Tri itératif	Tri basé sur un ou plusieurs parcours itératifs de la liste à trier
Tri récursif	Tri basé sur une méthode récursive
Tri en place	Tri qui n'utilise une quantité constante d'éléments en mémoire
Tri avec listes auxiliaires	Tri qui crée des listes auxiliaires pour réaliser le tri
Tri stable	Tri qui conserve l'ordre relatif des éléments de même clef

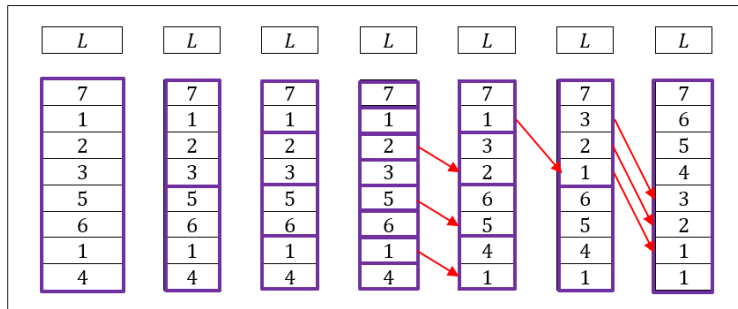
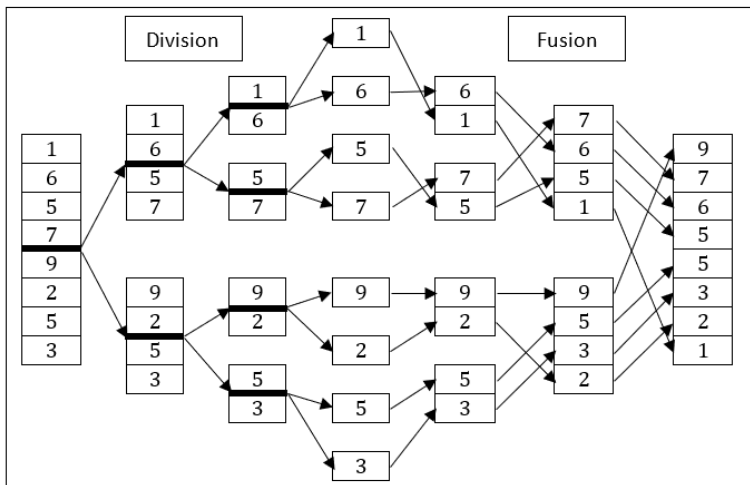
Python propose des fonctions « boîte noire » de tri	
En place	<code>L.sort()</code>
Avec listes auxiliaires	<code>Ll = sorted(L)</code>

Tri par insertion			
En place	On parcourt la liste L. Dès qu'un élément est inférieur au précédent, on échange les deux termes, et on répète cet échange autant de fois que nécessaire avec les termes d'en dessous		
			
Avec listes auxiliaires	On crée une liste dans laquelle on empile chaque terme de L du premier au dernier. A chaque ajout, on descend le terme empilé tant qu'il est inférieur à celui d'en dessous		
			
Complexité en temps	Meilleur des cas	$O\left(\sum_{i=1}^{n-1} 1\right)$	$O(n)$
	Pire des cas	$O\left(\sum_{i=1}^{n-1} (i-1)\right)$	$O(n^2)$
Stabilité	Dans les deux algorithmes ci-dessus, si on veille à ne descendre un terme que tant qu'il est supérieur strictement à celui d'en dessous, l'algorithme est stable		

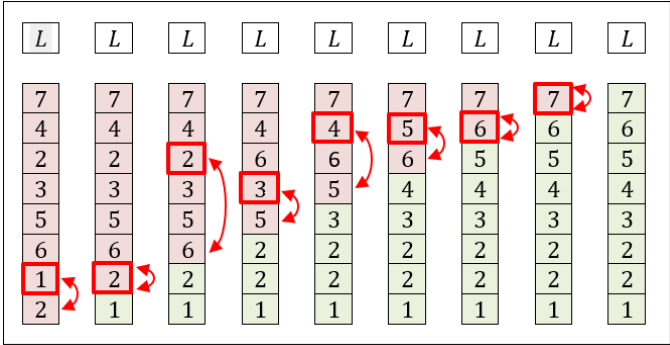
Dernière mise à jour	Informatique	Denis DEFAUCHY – Site web
30/03/2023	8 - Tris	Résumé

Tri rapide / Quick sort			
Un pivot doit être choisi. Naïvement, on prend la première valeur de la liste, mais il peut être intéressant de prendre une autre valeur, par exemple proche du centre de la liste si celle-ci est déjà presque triée (cf complexité)			
En place	<p>Considérer une portion de liste entre les indices i et j inclus. A la première itération, c'est la liste entière. Si cette sous liste contient un seul terme, ne rien exécuter, elle est déjà « triée ». Sinon :</p> <ul style="list-style-type: none"> - Choisir le pivot : Naïvement, le premier. Sinon, en choisir un et l'échanger avec la première valeur de la portion de liste traitée - Appeler p l'indice du pivot : $p = i$ - Définir un indice q qui au départ vaut p - Etudier chaque valeur de la sous liste étudiée pour un indice k entre les indices $i + 1$ et j inclus et procéder ainsi : <ul style="list-style-type: none"> o $L[k] \geq \text{Pivot} \rightarrow \text{RAS}$ o $L[k] < \text{Pivot}$: <ul style="list-style-type: none"> ▪ $q += 1$ ▪ $L[q], L[k] = L[k], L[q]$ - A la fin, on échange le pivot avec le terme à la position q : $L[p], L[q] = L[q], L[p]$ - Appeler récursivement cette procédure sur les « sous » listes de part et d'autre du pivot de la portion de liste étudiée <p>Remarques : A chaque étape, le pivot est définitivement placé. Dès que le pivot se trouve à l'indice milieu de la liste (milieu à définir selon parité de la taille de L...), ce pivot est la médiane de la liste ☺. Pour la médiane :</p> <p>Meilleur des cas : 1 seul auto-appel au rang $n/2$ avec une étape en $O(n)$, soit $O(n)$</p> <p>Pire des cas : des auto-appels 1 fois au rang $n-1$ avec $O(n)$ à chaque étape, soit $O(n^2)$</p>		
Avec listes auxiliaires	<p>Traiter les cas de base : Si la liste est vide, la renvoyer</p> <p>Choisir un élément de L appelé « pivot » (naïvement, première valeur)</p> <p>Créer les listes L_1 et L_2 telles que $\begin{cases} \forall i \neq 0, L_1[i] < \text{Pivot} \\ \forall i \neq 0, L_2[i] \geq \text{Pivot} \end{cases}$</p> <p>Appliquer récursivement le tri aux listes L_1 et L_2 pour obtenir deux listes L'_1 et L'_2 triées</p> <p>Renvoyer les listes combinées dans l'ordre : $[L'_1, \text{Pivot}, L'_2]$</p>		
Complexité en temps	Meilleur des cas	Auto-appel $\gamma > 1$ fois au rang n/γ avec $\alpha = 1 - \gamma = 2$	$O(n \ln n)$
	Pire des cas	Auto-appel 1 fois au rang $n-1$ avec $\alpha = 1$	$O(n^2)$
Stabilité	La version aux est stable si, lorsqu'un pivot possède des ex aequo, ceux-ci sont placés dans la liste des éléments supérieurs – La version en place ne l'est pas (cf. cours)		

Dernière mise à jour	Informatique	Denis DEFAUCHY – Site web
30/03/2023	8 - Tris	Résumé

Tri fusion			
En place	<p>Soit une liste L à trier.</p> <ul style="list-style-type: none">- Considérer une portion de liste dans l'intervalle d'indices Python $[i, j[$. A la première itération, c'est la liste entière.- Traiter le cas de base : Si $j = i + 1$, la portion de liste traitée ne contient qu'un terme, elle est déjà triée- Déterminer un indice milieu (gauche ou droite) m entre i et j- Appeler récursivement la fonction de tri fusion sur les deux portions de L dans les intervalles d'indices $[i, m[$ et $[m, j[$- En supposant que les portions de listes sur les deux intervalles $[i, m[$ et $[m, j[$ ont chacune été triées en place à l'étape précédente, appliquer une fusion ordonnée en place des termes de L entre i et j <p>Remarque : On pourra pour la fusion ordonnée, « descendre » de la partie $[m, j[$ les termes qui le nécessitent, dans la partie $[i, m[$, en décalant tous les autres termes « vers le haut » et en mettant à jour les indices des zones traitées</p>		
			
Avec listes auxiliaires	<p>Soit une liste L à trier :</p> <ul style="list-style-type: none">- Traiter le cas de base : Si L ne contient qu'un terme, elle est triée- Partager L en 2 listes L_1 et L_2 de tailles identiques (à 1 près)- Appliquer récursivement la procédure aux listes L_1 et L_2 pour les trier- En supposant que L_1 et L_2 ont été triées à l'étape précédente, les fusionner de manière ordonnée		
			
Complexité en temps	Meilleur des cas	Auto-appel $\gamma > 1$ fois au rang n/γ avec $\alpha = 1 - \gamma = 2$ (en supposant pour la version en place, que la fusion ordonnée est en $O(n)$)	$n \ln n$
	Pire des cas		
Stabilité	Dans les deux algorithmes ci-dessus, si dans la fusion ordonnée, on choisit le terme de gauche s'il y a des ex aequo, alors ce tri est stable		

Dernière mise à jour	Informatique	Denis DEFAUCHY – Site web
30/03/2023	8 - Tris	Résumé

Tri sélection		
Avec listes auxiliaires ou en place	<p>Soit une liste L à trier :</p> <ul style="list-style-type: none"> - Sélectionner le minimum de L et échanger ce minimum avec le premier terme de L (indice $i = 0$) - Sélectionner le minimum de L privée de son premier terme et échanger ce minimum avec le second terme de L - Etc. 	
	<p>Exemple en place</p> 	
Complexité en temps	$O\left(\sum_{i=0}^{n-2} (n-i)\right)$	$O(n^2)$
Stabilité	Lorsqu'il y a des exæquo, on garde leur ordre d'apparition en sélectionnant le premier minimum de L dans l'algorithme proposé	

Tri par comptage																																	
Avec listes auxiliaires ou en place	Soit une liste L de n entiers positifs à trier :																																
	<ul style="list-style-type: none">- Initialisation : Déterminer la valeur maximale m de la liste L et créer une liste D remplie de $m + 1$ valeurs nulles (chaque élément $D[i]$ représente le nombre d'apparition de la valeur i dans L)- Dénombrement : compter les apparitions de chaque terme de L par un parcours de la liste L en incrémentant de 1 l'élément $D[L[i]]$ – A la fin de cette étape, D contient le nombre d'apparition de chaque élément de L- Reconstruction : Créer une nouvelle liste contenant dans l'ordre, les éléments de L autant de fois qu'ils apparaissent, à l'aide de la liste D																																
	$L = [3, 2, 3, 4, 5, 6, 1, 2, 5, 3, 6, 7, 9, 2]$																																
	Initialisation		$m = 9$																														
	Dénombrement		<table><tr><td>i</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr><tr><td>$D[i]$</td><td>0</td><td>1</td><td>3</td><td>3</td><td>1</td><td>2</td><td>2</td><td>1</td><td>0</td><td>1</td></tr></table>										i	0	1	2	3	4	5	6	7	8	9	$D[i]$	0	1	3	3	1	2	2	1	0
i	0	1	2	3	4	5	6	7	8	9																							
$D[i]$	0	1	3	3	1	2	2	1	0	1																							
Reconstruction		$[1, 2, 2, 2, 3, 3, 3, 4, 5, 5, 6, 6, 7, 9]$																															
Complexité en temps	$O(2n + 2m)$										$O(n + m)$																						
Stabilité	Non concerné																																