ALGORITHME DE TRI

Vinod / Théo/ Nigson

Sommaire

Table des matières

Tri par Insertion

Fonctionnalité :

Commence sur le premier élément du tableau, et le compare avec l'élément suivant, si le premier élément du tableau est supérieur à l'élément suivant alors les deux éléments permutent.

Ainsi, nous savons que la partie gauche du tableau est déjà triée, donc la boucle ne repasse pas sur le côté gauche du tableau, elle reprend à la position suivante du dernier élément qui a été permuté et le compare avec tous les éléments précédents.

Si l'élément comparé est supérieur à un des éléments précédents, alors il sera l'élément suivant de cet élément, ce qui décalera d'une case toutes les valeurs suivantes du tableau.

3	7	2	6	5	1	4
3	7	2	6	5	1	4
2	3	7	6	5	1	4
2	3	6	7	5	1	4
2	3	5	6	7	1	4
1	2	3	5	6	7	4
1	2	3	4	5	6	7

Comme nous pouvons le voir, le côté gauche est toujours trié, et la position est stockée, ce qui permet de comparer l'élément suivant avec les éléments précédents.

Ex : Le premier élément comparé est 7, donc nous regardons si 7 est inférieure à l'élément précédent qui est 3, or ce n'est pas le cas donc il n'y a pas de permutation.

Puis, la boucle reprend sur l'élément 2 et le compare avec les éléments précédents, et ainsi de suite.

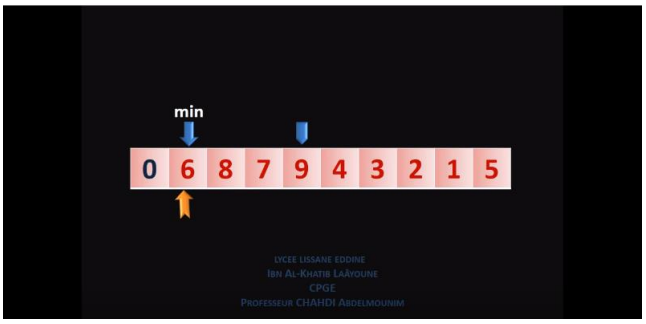
Tri par sélection

Fonctionnalité :

Commence sur le premier élément du tableau qui sera considéré comme la valeur minimum et la position est stockée dans une variable.

Le premier élément sera comparé aux éléments suivants, jusqu'à ce qu'une des valeurs suivantes soit inférieur, si elle est inférieure, alors elle deviendra le minimum, mais le tableau ne recommence pas sa boucle, il continue de comparé à la suite de la dernière position ou il a trouvé l'autre minimum.

S'il n'y a pas d'autre valeur minimum alors le dernier minimum pris en compte permutera avec le premier minimum de la boucle :



Ex : le minimum est 6, il va être comparé avec les valeurs suivantes, donc le prochain minimum sera 4, puis 3, puis 2 et 1, puis 1 permutera avec 6, car c'est le plus petit.

Puis la boucle de comparaison suivante, commencera à 8, et fera le même traitement.

Tri à bulle

Fonctionnalité :

Compare les éléments du tableau par groupe de deux, et les permutes si l'élément de gauche est supérieur à l'élément de droite (tout dépend de votre type de tri (petit -> grand) ou (grand -> petit)), puis il passe à la case suivante et compare les deux éléments suivants et ainsi de suite.

Ce tri permet de placer la plus grande valeur de votre tableau, à la fin du tableau (dans notre cas petit->grand).

A chaque nouveau tour de boucle, la boucle s'arrête à la taille du tableau – le nombre de tour de tableau qui a été effectué, ce qui permet de ne pas comparer les derniers éléments du tableau qui sont déjà triés.

	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]
	7	4	1	2	9	5
j = 1	4	7	1	2	9	5
j = 2	4	1	7	2	9	5
j = 3	4	1	2	7	9	5
j = 4	4	1	2	7	9	5
j = 5	4	1	2	7	5	9
j = 1	1	4	2	7	5	9
j = 2	1	2	4	7	5	9
j = 3	1	2	4	7	5	9
j = 4	1	2	4	5	7	9
j = 5	1	2	4	5	7	9

Le problème de ce tri, c'est qu'il revient au début du tableau pour chaque boucle et compare par bloque de deux, ce qui signifie, un nombre d'itération conséquent.

Tri de Shell

Fonctionnalité :

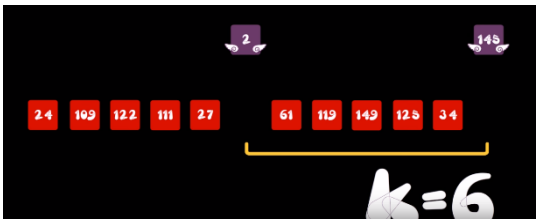
C'est un tri qui possède un écart de comparaison entre les éléments du tableau, cet écart permet de ranger une partie des plus petites valeurs du tableau vers la gauche, afin de simplifier le tri

Lors du premier tour de la boucle, l'écart est défini par la taille du tableau divisée par deux.

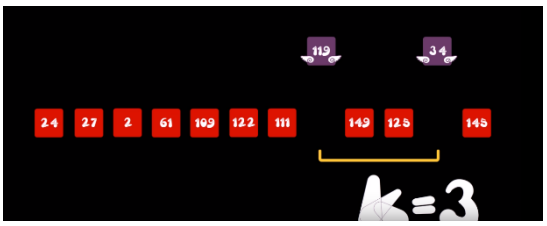
Ainsi, nous pouvons comparer les éléments grâce à cet écart, et si l'élément de droite est plus petit que l'élément de gauche, ils seront permutés (Il est comparé avec les éléments précédents pour voir si il n'y a pas plus petit en utilisant le même écart).

A chaque tour de boucle, cet écart est divisé par deux, afin de réduire l'écart de comparaison.

Tant que l'écart est supérieur à 0, alors cette procédure se répétera.



Comparaison avec un écart de 6 (Comme 2 n'est pas inférieure à 145, alors il n'y aura pas de permutation).



Puis l'écart sera divisé par deux et si l'élément de droite est plus petit que celui de droite, alors l'élément de droite sera aussi comparé avec les éléments précédents en respectant le même écart, afin de pouvoir le permuter.

Il est comparé (34) avec l'élément précédent et sera permuté car il est plus petit que 109

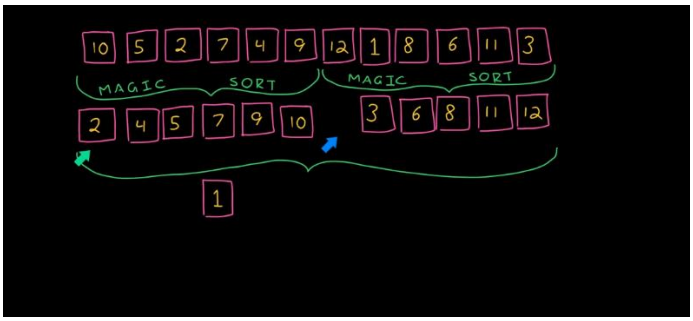
Tri Fusion

Fonctionnalité :

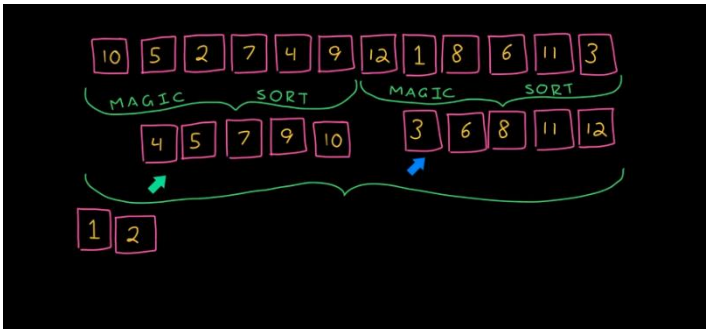
Ce tri, sépare le tableau en deux parties, généralement on prend la taille du tableau et on la divise par deux afin de savoir ou le séparer.

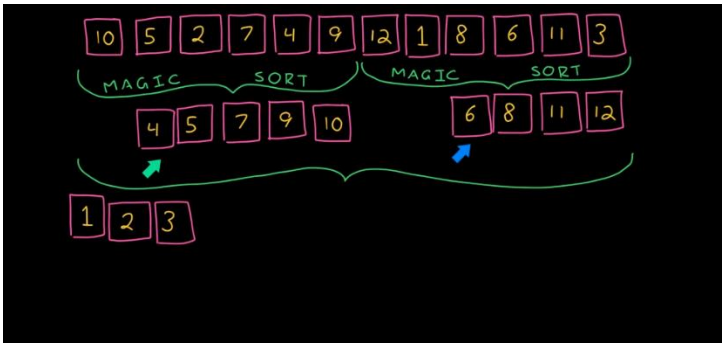
Une fois que le tableau est séparé en deux parties, on fait un tri dans chacun de ces tableaux, afin d'avoir deux tableaux déjà trié dans l'ordre croissant (Tout dépend de votre algo).

Puis, nous comparons un élément du tableau de gauche avec un élément du tableau de droite, et le plus petit des deux sera alors ajouté dans un nouveau tableau que je nommerais $newTab par exemple.



Puis l'élément qui était plus grand dans la comparaison précédente, sera comparé avec l'élément suivant du tableau qui a été comparé précédemment, et s'il est plus petit alors il sera rajouté au tableau $newTab, et ainsi de suite.





Tri Rapide

Fonctionnalité :

Dans ce type de tri, nous avons ce que l'on appelle un pivot.

Ce pivot est une référence de comparaison, car il sera comparé aux autres éléments du tableau afin de savoir, s'il faudra mettre l'élément comparé à gauche ou à droite du tableau.

Les éléments les plus petits doivent être placés, à gauche du pivot et les plus grand à droite du pivot.

Lors du premier tour de boucle, le pivot commence sur le premier élément de votre tableau et va comparer chaque élément avec ce pivot, s'il est plus petit que le pivot alors il sera mis dans le tableau $left, sinon dans le tableau $right.

Puis, nous utilisons la récursive afin d'obtenir plusieurs tableaux, ne comportant qu'un élément, or ils sont déjà triés.

Enfin, nous utilisons array_merge() afin de réunir tous ces tableaux, en un seul tableau qui sera déjà trié, en commençant par le tableau de gauche, le tableau du pivot et le tableau de droite.

Tri à peigne

Le tri à peigne possède aussi un écart de comparaison, qui est calculé en divisant la taille du tableau par 1.3, lors du premier tour de boucle.

Puis, à chaque tour de boucle l'ancien écart est divisé par 1.3, ce qui permet de réduire l'écart de comparaison.

Comparaison : Si l'élément de gauche est plus grand que l'élément de droite, alors les éléments seront permutés.

L'écart de comparaison, permet de ranger les valeurs les plus faibles du tableau vers la gauche (Cela dépend si vous triez dans l'ordre croissant ou décroissant).

Le tri se termine lorsque la variable $swapped est égale à FALSE, ce qui signifie qu'il n'y a eu aucune permutation lors de ce tour, donc le tableau est bien trié. Elle prend uniquement la valeur TRUE lorsqu'il y a eu une permutation entre deux éléments du tableau.

Conclusion

Nous avons pu constater qu'il y avait deux tris qui était efficace mais cela dépendait des cas :

- Fusion
- Rapide