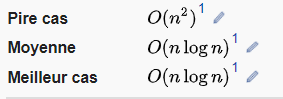
**Tri rapide:** fondé sur la méthode de conception divisé pour régné, La méthode consiste à placer un élément du tableau (appelé pivot) à sa place définitive, en permutant tous les éléments de telle sorte que tous ceux qui sont inférieurs au pivot soient à sa gauche et que tous ceux qui sont supérieurs au pivot soient à sa droite.

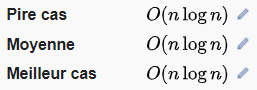
Cette opération s'appelle le partitionnement. Pour chacun des sous-tableaux, on définit un nouveau pivot et on répète l'opération de partitionnement. Ce processus est répété récursivement, jusqu'à ce que l'ensemble des éléments soit trié.

 Quasi-linaire et quadratique dans le pire des cas.

**Tri fusion :** basé sur la technique divisé pour régné, qui consiste á réunir 2 listes triées en une seule :

L'algorithme est naturellement décrit de façon récursive.

1. Si le tableau n'a qu'un élément, il est déjà trié.
2. Sinon, séparer le tableau en deux parties à peu près égales.
3. Trier récursivement les deux parties avec l'algorithme du tri fusion.
4. Fusionner les deux tableaux triés en un seul tableau trié.

Quasi-linéaire dans toutes les cas.

**Tri par insertion :** il considère le tableau en 2 parties, partie trié, partie non trié, donc á chaque fois il insert un élément de la partie non trié á sa bonne place á l’intérieur de la partie triée, jusqu’à ce que tous le tableau soit trié á la fin 🡪 complexité quadratique.

**Tri á bulles :** il consiste á comparer répétitivement les éléments consécutifs d’un tableau jusqu’à ce qu’il soit trié 🡪 complexité quadratique.

**Tri par sélection :** consiste á trouver le plus petit élément du tableau et le mettre dans la première case, puis de trouvé le 2 plus petits élément et le mettre dans la deuxième, on répète cette technique jusqu’à ce que tous le tableau soit trié 🡪 complexité quadratique.

**Recherche dichotomique** : la recherche se fait dans un tableau trié, en comparant l'élément avec la valeur de la case au milieu du tableau :

* Si les valeurs sont égales, la tâche est accomplie.
* Si l’élément plus grand que la valeur du milieu, on justifier le début de tableau par le milieu + 1.
* Si l’élément plus petit que la valeur du milieu, on justifier la fin de tableau par le milieu – 1.
* Si la fin plus petit que le début de tableau on arrête l’opération et que la valeur n’existe pas dans le tableau.

**Liste chainée :** est une collection des éléments de même type de taille arbitraire dont chaque élément possède un pointeur sur l’élément suivant, par conséquent les éléments sont dispersées dans la mémoire.

**La pile :** est une structure de données fondé sur le principe dernier arrivé premier sortie (LIFO), peut être implémenter soit avec une liste chainé (empiler : ajouter en tête, dépiler : supprimer en tête), soit avec un tableau.

**La file :** est une structure de données qui est fondé sur le principe premier arrivé dernier servie (FIFO), peut être implémenter á l’aide d’une liste chainé (enfiler : ajouter fin, défiler : supprimer tète), ou bien á l’aide d’un tableau.

**Arbre :** est une structure de donnée, composée d’un ensemble de nœud, chaque nœud contient une information spécifique et des pointeurs vers des autres nœuds.

**Profondeur d’un nœud** : le nombre des nœuds qui sépare cette nœud avec la racine.

**Hauteur :** la profondeur du nœud le plus loin de racine, profondeur maximal.

**Parcours préfixe :** racine, arbre gauche, arbre droit.

**Parcours infixe :** arbre gauche, racine, arbre droit.

**Parcours suffixe :** arbre gauche, arbre droit, racine.

**Parcours en largeur :** parcours niveau par niveau.

**Arbre de binaire de recherche:** est un arbre dont la valeur de chaque nœud est supérieure á la valeur de son fils gauche est inferieur á la valeur de son fils droit.

* **La hauteur d’un arbre de recherche** ayant n nœuds est toujours inferieur á log(n).
* **La complexité** de L’insertion, la rechercher, et la suppression dans un arbre binaire de recherche dans le cas moyen est de O(log(n)), dans le pire des cas est de O(n) (l’arbre est complétement déséquilibre, ressemble á une liste chainé)
* **Un arbre binaire de recherche est dit complet** si tous les niveaux de l'arbre sont remplis, sauf éventuellement le dernier, sur lequel les nœuds sont à gauche.
* **Un arbre binaire parfait** est un arbre complet dont toutes les feuilles sont à la même hauteur (le dernier niveau est complètement occupé).
* **Un arbre binaire est dit dégénéré** si chacun de ses nœuds a au plus un fils.
* **Un arbre binaire est équilibré** si tous les chemins de la racine aux feuilles ont la même longueur.

**Recherche :** on effectue une comparaison de l’élément avec la valeur de racine :

* Si l’élément est supérieur á la valeur de racine, on effectue la recherche sur le sous arbre droit.
* Si l’élément est inferieur á la valeur de racine, on effectue la recherche sur le sous arbre gauche.
* S’ils sont égaux on retourne la racine de l’arbre où on est arrêté.
* Si on est arrivé á un branche null, donc la valeur n’existe pas dans l’arbre.

**Insertion :** insertion d’un nouvel élément dans un arbre binaire de recherche.

* Si l’arbre est vide, donc l’élément sera la racine d’un nouvel arbre.
* Si l’élément est supérieur á la valeur de racine, on insert sur le sous arbre droite.
* Si l’élément est inferieur á la valeur de racine, on insert sur le sous arbre gauche.
* S’ils sont égaux, on revient sans insertion.

**Élément maximum dans un ABR** : c’est le dernier nœud qui n’a pas de fils droit et qui se trouve extrêmement á droit de l’arbre.

**Suppression de l’élément maximum de l’arbre :** tout d’abord on se déplace vers l’élément maximum de l’arbre, ensuite on le remplace par son fils gauche.

**Suppression d’un élément dans un ABR :** la suppression par se fait par un élément á supprimer :

* Si le nœud est une feuille, la suppression est immédiate.
* Si le nœud á un fils, on le remplace par ce fils.
* Si le nœud á un fils, on choisit de le remplacer, soit par l’élément max de son fils gauche, soit par l’élément min de son fils droit.

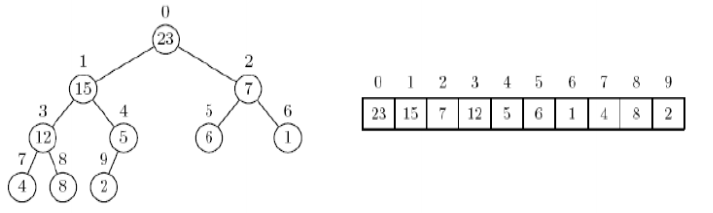
**Arbre équilibrés au sens des hauteurs (AVL) :** est un arbre binaire de recherche tel que les hauteurs de ses 2 fils se diffèrent au plus de 1.

Á chaque insertion en vérifier est que l’arbre est équilibrés si non on précède á des rotations pour l’équilibré.

**Arbre maximiers (TAS) :** est un arbre binaire parfait tel que la clé de chaque nœud est supérieur ou égal aux clés de tous ses fils, l’élément max donc se trouve á la racine.

Les tas sont représentés par un tableau ou la racine si la valeur de la case d’indice 0, un nœud se trouve á l’indice i :

* Son fils gauche se trouve au nœud de l’indice 2i + 1.
* Son fils droit se trouve au nœud de l’indice 2(i + 1).
* Son père se trouve á l’indice (i-1) /2.



**Insertion :** on insert l’élément á la dernière case du tableau, et on compare si sa valeur est plus grand de la valeur son père qui se trouve á la case (i-1) /2 si oui on procède á une permutation, et ainsi de suite jusqu’à ce que sa valeur est n’est plus grand que de son père ou on est arrivé á la case d’indice 0.

**Suppression du maximum :** on échange la racine (maximum) avec le dernier élément du tableau, et on fait descendre l’élément dernier de la racine en le permutant avec le grand de ses 2 fils, et en répète cette itération jusqu’à ce qu’on arriver á la fin du tableau.