1. **Tri par insertion (Insertion Sort)**

Le tri par insertion s’inspire de la façon dont les cartes sont triées dans une main. À chaque étape, un élément du tableau est extrait et inséré dans sa position correcte parmi les éléments déjà triés. Cela implique de déplacer les éléments plus grands d’un pas à droite pour faire de la place. Progressivement, la portion triée du tableau s’étend de gauche à droite, jusqu'à ce que tous les éléments soient triés. Cet algorithme est particulièrement efficace pour des tableaux déjà partiellement triés.

1. **Tri à bulles (Bubble Sort)**

Le tri à bulles fonctionne en comparant chaque paire d'éléments adjacents d'un tableau et en échangeant leurs positions si l'ordre est incorrect. À chaque passage, l'élément le plus grand non trié "remonte" vers sa position finale à la fin du tableau, comme une bulle d'air remontant à la surface. Le processus se répète pour les éléments restants, réduisant la plage des comparaisons à chaque itération. L'algorithme s'arrête lorsque aucun échange n'est nécessaire, garantissant ainsi que le tableau est trié.

1. **Tri rapide (Quick Sort)**

Le tri rapide choisit un élément comme pivot, puis partitionne le tableau en deux sous-tableaux : l’un contenant les éléments inférieurs au pivot, et l’autre contenant les éléments supérieurs. Le pivot est alors à sa position finale. Cette partition est répétée récursivement sur les deux sous-tableaux jusqu'à ce qu'ils soient réduits à une seule unité. Sa stratégie de "diviser pour régner" en fait l’un des algorithmes les plus performants pour le tri général.

1. **Tri par sélection (Selection Sort)**

Le tri par sélection identifie en chaque itération le plus petit élément parmi les éléments non triés et l’échange avec l’élément à la position courante. Ainsi, le tableau se construit progressivement en plaçant les plus petits éléments dans l’ordre croissant à partir du début. Une fois un élément positionné, il n’est plus touché, et l’algorithme poursuit sur la partie restante jusqu'à ce que tout soit trié.

1. **Tri de Shell ou necore Shell Sor**

Le tri de Shell améliore le tri par insertion en comparant et en échangeant des éléments éloignés les uns des autres dans un tableau. L’algorithme commence par un "écart" important entre les éléments comparés, puis réduit cet écart progressivement. À mesure que l'écart diminue, le tableau devient de plus en plus trié, jusqu'à ce qu'il reste un écart de 111, équivalent au tri par insertion sur l’ensemble du tableau.

1. **Tri par tas // Heap Sort**

Le tri par tas utilise une structure de données appelée tas (heap) pour organiser les éléments. D'abord, le tableau est transformé en un tas binaire max, où le plus grand élément est à la racine. Cet élément est ensuite échangé avec le dernier élément du tableau, et la taille du tas est réduite. Le processus est répété en réajustant le tas jusqu'à ce que tous les éléments soient triés. Cet algorithme garantit un tri efficace sans mémoire supplémentaire.

1. **Tri fusion (Merge Sort)**

Le tri fusion adopte une approche "diviser pour régner". Il divise le tableau en deux moitiés jusqu'à ce que chaque sous-tableau contienne un seul élément. Ensuite, il fusionne ces sous-tableaux de manière ordonnée pour reconstruire progressivement le tableau trié. Cette méthode garantit un tri stable et est particulièrement adaptée aux grandes structures de données grâce à sa complexité logarithmique.

1. **Tri comptage Counting Sort**

Le tri comptage est une méthode non comparative qui compte les occurrences de chaque élément dans une plage donnée. Ces comptes sont utilisés pour déterminer les positions finales des éléments dans le tableau trié. Les éléments sont ensuite placés dans un tableau de sortie dans l’ordre. Cet algorithme est efficace pour des données discrètes dans une plage limitée, mais consomme de la mémoire proportionnelle à la plage des valeurs.

1. **Tri par casiers ou encore Bucket Sor**

Le tri par casiers distribue les éléments dans plusieurs "casiers" ou sous-listes en fonction de leur valeur. Chaque casier est trié individuellement à l'aide d'un algorithme simple, puis les casiers sont concaténés pour former le tableau trié. Cet algorithme est particulièrement efficace pour des données uniformément réparties sur une plage connue.

1. **Tri radix encore appelé en anglais radix sort**

Le tri radix traite les nombres en les triant chiffre par chiffre, généralement du moins significatif au plus significatif. Chaque étape utilise un tri stable, comme le tri comptage, pour regrouper les nombres en fonction d’un chiffre donné. Cette méthode est idéale pour les ensembles de données ayant une longueur fixe, comme des nombres ou des chaînes de caractères de longueur limitée.