**CER 1.8: Tris et recherche**

**Mots-clés :**

* Algorithme de tri : Suite logique d’informations permettant de classer suivant un certain ordre une suite de nombre, caractère ou chaine de caractère.
* Complexité quadratique = Complexité en temps ou spatiale (*Cf plus bas*), calculé en fonction du temps mis lors de l’exécution d’un programme et permet de comparer efficacement deux programmes, de tris par exemple.
* Tri à bulle : *Cf plus bas*
* Tri fusion : *Cf plus bas*
* Tri par insertion : *Cf plus bas*
* Tri successif : *Cf plus bas*
* Tableau de structure : Tableau de type de variables « structures », pouvant ainsi contenir plusieurs types de variables à l’intérieur de ce tableau.
* Listing : Lignes de codes d'un programme informatique ou d'un fichier de données dans un format lisible par un humain.

**Contexte :**

L’ouragan Irma s’approche des côtes françaises et on souhaite faire un listing des habitants.

**Contraintes :**

* Le temps (1 ou 2 jours)
* Faire 2 tris successifs
* + de 500 000 personnes

**Problématiques :**

* Comment lister les 500 000 personnes ?
* Comment trier les 500 000 personnes ?
* Quelle est la structure d’un tri ?
* Comment savoir si une personne a été déjà été triée ou non ?
* Quelle sont les différences entre tous les types de tri ?
* Que signifie la complexité d’un tri et comment l’évaluer ?
* Comment identifier la collectivité d’une personne en fonction d’un nom et d’un prénom

**Généralisation :**

Création d’un système de tri.

**Pistes de solution :**

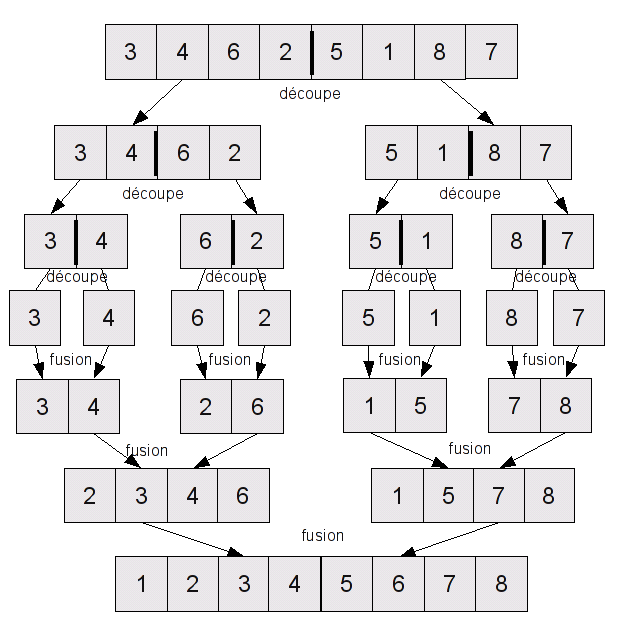
* On doit utiliser des structures
* Faire un tableau
* Faire plusieurs tableaux, puis les regrouper dans un seul les uns après les autres pour finaliser une liste
* 1 personne = 1 numéro ; plus ce numéro est élevé, plus il est proche du littoral
* Utiliser des tableaux spéciaux
* La complexité dépend du nombre de facteurs utilisés lors du tri

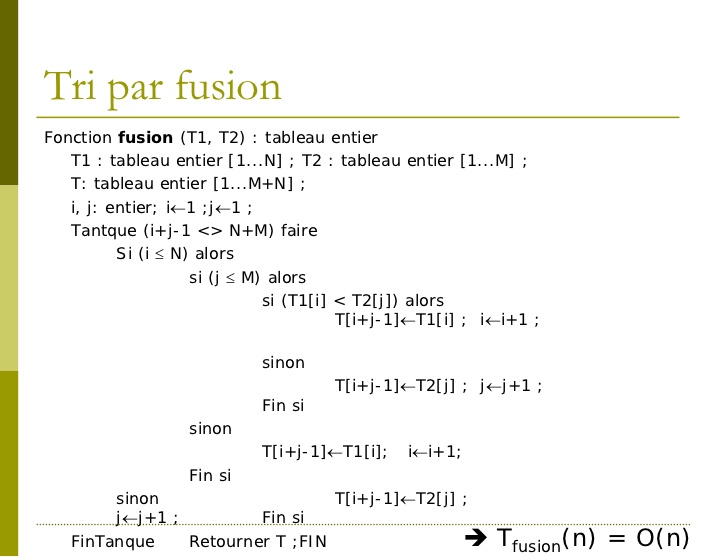
**Plan d’action :**

* Etudier le tri
* Etudier la complexité
* Choisir le tri le plus adapté
* Faire l’algorithme de tri
* Elaborer le programme de tri
* Créer un algorithme de recherche
* Corbeille d’exo

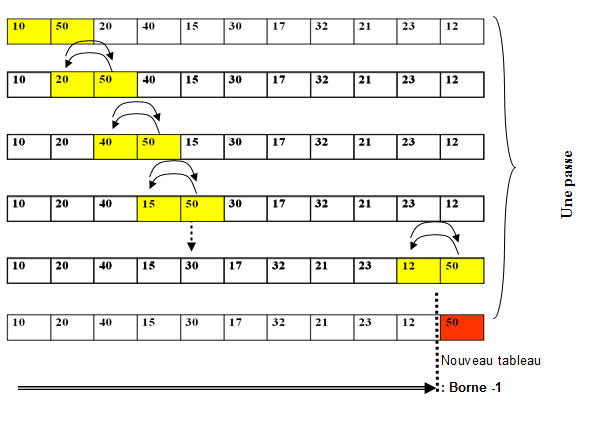
Différents types de tris, explication de manière concrète puis algorithmique:

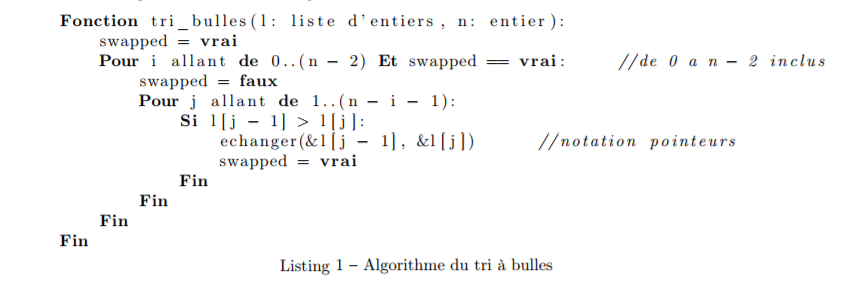
1. *Tri par fusion :*



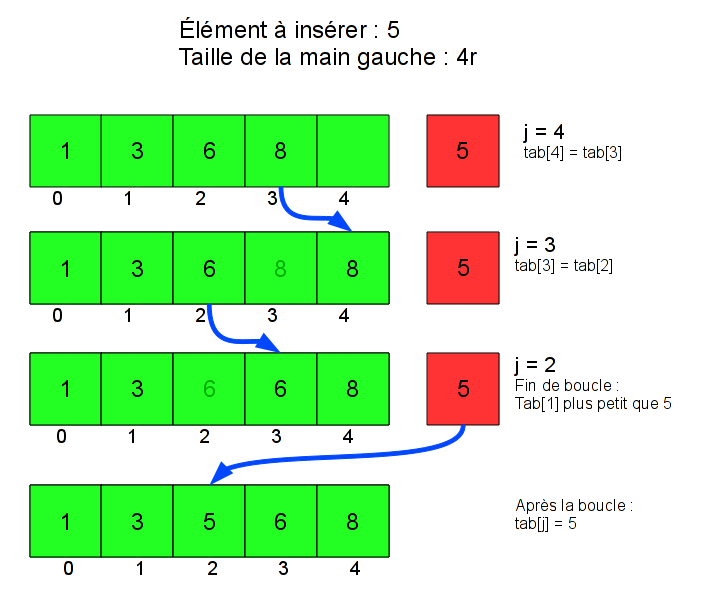


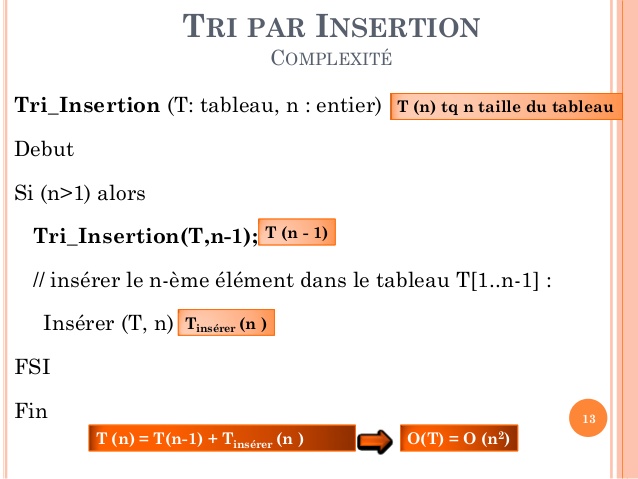
1. *Tri par bulles*



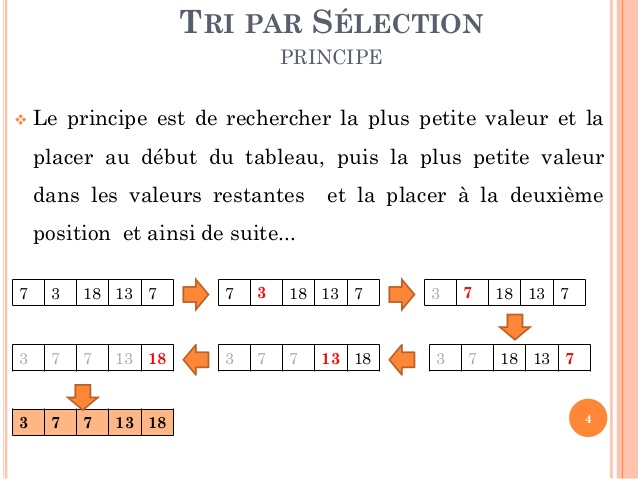


*3) Tri par insertion :*

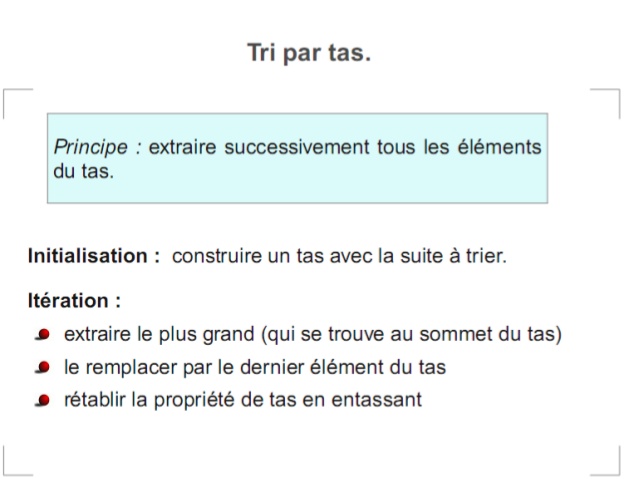




*4) Tri par selection :*



*5) Tri par tas :*



2) Explication de la notion de **complexité**: <https://openclassrooms.com/courses/algorithmique-pour-l-apprenti-programmeur/la-notion-de-complexite> 🡪 permet de comparer deux programmes ou bout de programme en fonction du temps pour un nombre d’entrée N ( pouvant même tendre vers l’infini dans certain cas 🡪 mesure asymptotique) ou d’espace.

Différents types de complexité :

*Spatiale* : Utilisation d’un minimum de **variables** ( 🡪 Moins de mémoire utilisée possible)

*Temporelle* : Exécution la plus **rapide**.

3) Dans notre cas le tri par **fusion** semble le plus intelligent à utiliser, car sur un nombre important d’entrées N (+500 000) il se révèle être le plus efficace en termes de rapidité.

4) Revision des **structures, tableaux 1D et ouverture de fichiers** en langage c, afin de pouvoir lire et stocker les informations nécessaires aux tris.

Téléchargement du code source initial ( <https://moodle-exia.cesi.fr/mod/folder/view.php?id=11493> ) et modification afin d’y trier les informations recueillies à l’aide du tri fusion. 🡪 Réalisation de **tableaux de structures** afin de pouvoir trier toutes les informations de manière efficace.