# Algorithmique et structures de données

Loïc Demange
loic.demange@etud.univ-paris8.fr

06 novembre 2020

# TP précédent

Objectifs du TP :

- Créer une fonction qui prend en paramètre une taille et renvoie un tableau d'entiers de cette taille.
- Créer une fonction qui prend en paramètre un tableau d'entiers et une taille, et renvoie un autre tableau avec les mêmes valeurs mais triées par ordre croissant (un algorithme naïf suffira).

# TP précédent

**Créer une fonction qui prend en paramètre une taille et renvoie un tableau d'entiers de cette taille.**

```
#include <stdio.h>
#include <stdlib.h> // Bibliothèque contenant malloc
int* creer_tab(int n)
{
    int* tab = malloc(n * sizeof(int));
    return tab;
}
int main()
{
    int *tab = creer_tab();
    for(int i = 0; i < 10; i++)
        tab[i] = i;

    for(int i = 0; i < 10; i++)
        printf("%d ", tab[i]);
    printf("\n");

    free(tab); // On libère la mémoire
    return 0;
}
```

Ici, on est obligé d'allouer dynamiquement la mémoire du tableau, pour éviter qu'il ne soit désalloué à la sortie de la fonction.
Il ne faut pas oublier de faire appel à **free** pour éviter les fuites mémoires.

# TP précédent

**Créer une fonction qui prend en paramètre un tableau d'entiers et une taille, et renvoie un autre tableau avec les mêmes valeurs mais triées par ordre croissant (un algorithme naïf suffira).**

```
#include <stdio.h>
#include <stdlib.h>
int* tri_tab(int* tab, int n)
{
    int* tab_trie = malloc(n * sizeof(int));
    for(int i = 0; i < n; i++) // On recopie le tableau
        tab_trie[i] = tab[i];

    for(int i = 0; i < n; i++)
    {
        for(int j = 1; j < n; j++) // On reparcourt le tableau
        {
            // Si la case d'avant est plus grande que la case actuelle, on échange les positions
            if(tab_trie[j - 1] > tab_trie[j])
            {
                int temp = tab_trie[j];
                tab_trie[j] = tab_trie[j - 1];
                tab_trie[j - 1] = temp;
            }
        }

    }
    return tab_trie;
}
```

Plusieurs algorithmes sont possibles, celui-ci est le tri à bulles.

# TP précédent

**Créer une fonction qui prend en paramètre un tableau d'entiers et une taille, et renvoie un autre tableau avec les mêmes valeurs mais triées par ordre croissant (un algorithme naïf suffira).**

```
int main()
{
    int tab[10] = {9,8,7,6,5,4,3,2,1};
    int* tab_t = tri_tab(tab, 10);

    for(int i = 0; i < 10; i++)
        printf("%d ", tab_t[i]);

    free(tab_t); // On libère la mémoire
}
```

Voici un **main** qui appelle correctement **tri_tab** et qui libère bien la mémoire.

# Projet : Tri fusion

L'idée du tri fusion est que pour trier un tableau, on peut trier chaque moitié du tableau puis les fusionner.
On peut donc appliquer ce concept récursivement en subdivisant le tableau en deux à chaque itération, en ne remontant uniquement que lorsque le tableau ne possède plus qu'une seule valeur.

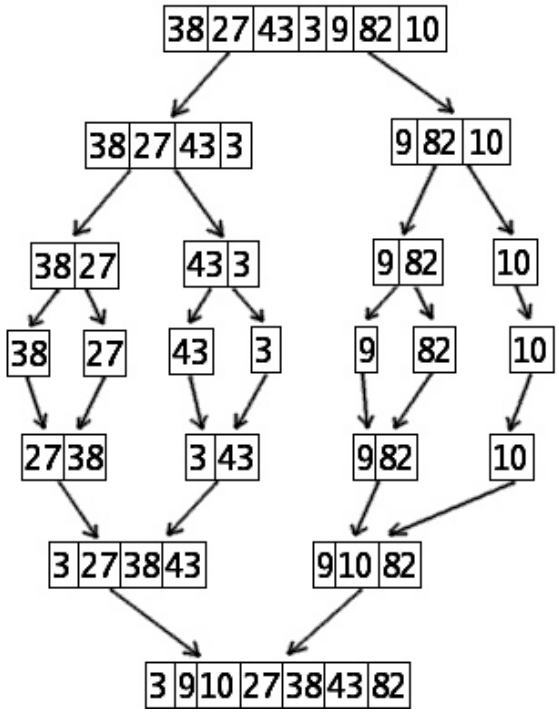


Figure: Tri d'un tableau de 7 éléments (wikipédia)

# Projet : Tri fusion

Le projet consiste en :

- Créer une fonction qui prend en paramètre un tableau d'entiers et une taille, et renvoie le tableau trié par ordre croissant à l'aide du tri fusion.
- Écrire un document expliquant précisément le fonctionnement du tri fusion, en énonçant et **justifiant** sa complexité temporelle.

Le code doit contenir tout ce qu'on a vu jusqu'à présent : de la dichotomie, de la récursivité et des tableaux.

Il peut se réaliser seul.e, ou de préférence, à deux, et est à rendre avant le 20 novembre 2020 à 15h (heure du cours).

Les recherches sur internet sont encouragées, le but étant non pas de réinventer l'algorithmique mais d'en comprendre le concept et l'implémenter.
Recopier du code sans le comprendre sera très vite repéré et ne vous apportera rien.