

ALGORITHMIQUE ET STRUCTURES DE DONNÉES

TP 2. Structure de données linéaires et tris

27 Janvier 2026

1 File de priorité par liste doublement chaînée

Le but de cet exercice est d'implémenter une file de priorité à l'aide d'une liste doublement chaînée. Pour cela vous complétez le fichier `file_priorite.cpp`.

Structure de liste double chaînée Commencer par implémenter les 4 fonctions de la structure `Liste` (définie dans `file_priorite.h`) :

- `vide` : qui retourne un booléen indiquant si la liste est vide.
- `ajouter_fin` : qui ajoute un élément à la fin de la liste.
- `ajouter_trie` : qui ajoute un élément à la bonne position dans une liste triée.
- `retirer` : qui retire une cellule de la liste.

Pour la fonction `ajouter_trie`, vous devrez parcourir les cellules de la liste jusqu'à trouver l'emplacement où insérer la nouvelle valeur.

Implémentation de files de priorité Le fichier *header* `file_priorite.h` définit deux files de priorité utilisant une `Liste`. Ces deux structures sont `FilePrioriteA` et `FilePrioriteB`. Vous pouvez constater que les méthodes `ajouter` sont déjà définies pour ces deux structures. `FilePrioriteA` utilise `ajouter_fin` tandis que `FilePrioriteB` utilise `ajouter_trie`. Il vous reste alors à implémenter la méthode `retirer` dans `file_priorite.cpp`. Pour `FilePrioriteB`, vous utiliserez le fait que la liste soit triée.

Tester Une fois implémentée, exécuter le programme et observer les différences de temps d'insertion et de suppressions entre les différentes files de priorités.

Les files de priorité sont testées en ajoutant 10 000 valeurs à la file (correspond au temps d'insertions), puis dans un second temps, toutes les valeurs de la file sont dépliés (correspond au temps de suppressions). Les temps correspondent-ils à ce que vous attendiez ?

Une dernière file `FilePrioriteC` est présente. Celle-ci utilise une `std::priority_queue` de la librairie standard de C++. Observer les temps faibles à la fois pour l'insertion et la suppression. Une `std::priority_queue` est un tas binaire, une structure qui permet l'insertion et la suppression du plus grand élément en $\mathcal{O}(\log(n))$.

2 Implémentation de tris

Le but de cet exercice est d'implémenter plusieurs fonctions de tri vu en cours. Pour cela vous complétez le fichier `tri.cpp`. Vous aurez trois tris à implémenter :

- Le tri par insertion.
- Le tri rapide.
- Le tri fusion. Vous commencerez par implémenter la fonction `fusion` qui fusionne deux tableaux triés L et R dans un seul tableau T . Afin d'utiliser cette fonction `fusion` dans `tri_fusion` vous pourrez être amenés à copier un tableau dans une nouvelle zone de la mémoire allouée. L'allocation d'un tableau de taille n se fait via :

```
int *tmp = new int[n];
```

N'oubliez pas, ensuite, de désallouer la mémoire :

```
delete[] tmp;
```

