Feuille de travaux dirigés n^o 4
Structures de données

Exercice 4.1 — Tri par base

On considère T un tableau **trié** de n entiers.

1. Faites un tri par base (10) du tableau suivant 122, 12, 22, 342, 111, 731, 428, 513, 23, 5.
2. Donnez les avantages et les inconvénients des tris par base suivant que l'on utilise la base 10, la base 2 et la base 256.

Exercice 4.2 — Complexité des Algorithmes de Tri

Pour chacun des cas suivants donnez l'algorithme de tri qui sera certainement le plus efficace

1. on a beaucoup de nombre entre 1 et 100
2. on a n nombres entre 1 et n
3. on a peu de nombres entre 1 et 1 milliard
4. on a très peu de nombres
5. on a beaucoup de nombres entre 1 et 1 milliard

Exercice 4.3 — Queue de priorité

En informatique, une queue de priorité est un type abstrait élémentaire sur laquelle on peut effectuer trois opérations :

— insérer un élément ;
— lire puis supprimer l'élément ayant la plus grande clé ;
— tester si la queue de priorité est vide ou pas.

On ajoute parfois à cette liste l'opération "augmenter la clé d'un élément"
On insère successivement les éléments dont les clés valent 3, 8, 4, 5, 2, 7. Puis on lit et supprime deux fois de suite l'élément ayant la plus grande clé. Ensuite, on insère les éléments dont la clé est 5 et 1. Enfin, on extrait deux fois l'élément ayant la plus grande clé.

1. Détaillez le fonctionnement de ces opérations en donnant notamment la valeur des clés des éléments supprimés.
2. Quelle est la complexité de ces opérations si vous décidez de conserver les éléments sous la forme d'un tableau ? d'un tableau dont vous maintenez le tri ?
3. Comment pouvez-vous maintenir le tri de façon efficace si un nouvel élément est introduit et si la valeur d'une clé est augmentée ?
4. Proposez une structure de données pour gérer ces opérations.