TD ITC n°2

Opérations sur tableaux triés

On explore dans ce TD un certain nombre d'opérations intéressantes sur les tableaux de nombre triés. Sauf mention contraire, les tableaux considérés sont triés, s'appellent `tabt` pour « tableau trié » et ont pour longueur n. Les tableaux sont représentés dans le code par des listes Python.

I. Préparation

1 – Récupérer le module `TD02_module.py` sur cahier de prépa, puis ouvrir un **nouveau fichier python** dans le même dossier que le module. Dans ce nouveau fichier, écrire l'instruction

```
import TD02_module as mod
```

puis utiliser la console pour créer un tableau trié :

```
In [1]: test = mod.tableau_trié(10, 6)
In [2]: test
Out[1]: [-5, -3, -3, -2, -1, 0, 0, 4, 5, 6]
```

On peut accéder à la documentation de cette fonction avec `help(mod.tableau_trié)`.

II. Plus petit, grand etc

2 – Écrire une fonction `minimum_tabt(tabt: list) -> int` qui prend en paramètre un tableau `tabt` et renvoie le maximum du tableau **en complexité $\mathcal{O}(1)$**.

3 – Écrire une fonction `maximum_tabt(tabt: list) -> int` qui prend en paramètre un tableau `tabt` et renvoie le maximum du tableau **en complexité $\mathcal{O}(1)$**.

4 – Écrire une fonction `médiane_tabt(tabt: list) -> int` qui prend en paramètre un tableau `tabt` et renvoie la médiane du tableau **en complexité $\mathcal{O}(1)$**.

III. Vérification des entrées

Dans cette section uniquement, **dont on ne sait pas si le tableau fourni est trié**.

5 – Écrire une fonction `est_trié(tab: list) -> bool` qui prend en paramètre un tableau `tab` et renvoie `True` s'il est trié, `False` sinon. La complexité doit être en $\mathcal{O}(n)$.

IV. Création de tableaux triés

Dans cette section, **les tableaux en entrée sont à nouveau supposés triés**.

6 – Écrire une fonction `insertion_tabt(tabt: list, x: int)` qui prend en paramètres un tableau trié `tabt` et un nombre x et rajoute dans `tabt` l'élément x à la bonne position pour que le tableau reste trié. La fonction pourra renvoyer `None` ou ne pas présenter d'instruction `return`. La complexité doit être $\mathcal{O}(n)$.

7 – Écrire une fonction `fusion(tabt1: list, tabt2: list)` qui prend en paramètres deux tableaux triés `tabt1` et `tabt2`, et créé et renvoie un grand tableau **trié** contenant la réunion de `tabt1` et `tabt2`. La complexité doit être $\mathcal{O}(n)$, où $n = n_1 + n_2$ est la taille du tableau final.