Implémentation des tris

juillet 2020

1 Tri rapide (bis)

Exercice 1

Reprenez et modifiez votre implémentation du tri rapide commencée lors d'une séance précédente. Il s'agit ici d'exploiter l'algorithme du drapeau hollandais pour :

- *effectuer* ***en place*** *la partition du tableau en valeurs inférieures et supérieures au pivot ;*
- *ne plus utiliser la concaténation pour rassembler les sous-tableaux triés ;*
- *comme tout le tri s'effectue dans un même tableau, votre fonction récursive devra prendre en arguments deux indices indiquant le segment du tableau qui est à trier.*

2 Tri par fusion

Exercice 2

Implémentez un tri par fusion de la façon la plus simple et lisible possible. Vous pouvez utiliser les facilités de Python : tranchage de liste, recopie de liste, etc.

Exercice 3

Améliorez le programme précédent pour éviter de réallouer un nouveau tableau temporaire à chaque appel de `fusion` *:*

- *Toutes vos fonctions devront prendre en argument supplémentaire la liste temporaire ; elles ne seront jamais appelées directement par l'utilisateur.*
- *La fonction principale* `tri_fusion` *se contente de créer la liste temporaire (une copie de la liste à trier, par exemple), puis de faire le premier appel à la fonction auxiliaire qui prend cette liste temporaire en argument.*

Exercice 4

Améliorez encore le programme précédent pour que la fusion ne fasse qu'une seule recopie des éléments de la portion à fusionner : on jouera donc sur les deux listes reçues en arguments, l'une servant à recevoir les données et l'autre à construire le résultat.

3 Tri par tas (très facultatif)

Pour ce TP, on considérera des tas dans lesquels c'est la valeur *minimale* qui est placée à la racine.

Exercice 5 (Structure de tas)

Programmer les fonctions suivantes, dans lequelles l'argument `T` *est un tas représenté sous forme d'une liste Python :*

1. `tas_vide()`
   renvoie un nouveau tas ne contenant aucun élément.
2. `inserer(x, T)`
   modifie `T` *pour y insérer la valeur* `x`*, de façon à conserver une structure de tas.*
3. `minimum(T)`
   renvoie l'élément minimal de `T`*, sans modifier ce dernier.*
4. `extraire_minimum(T)`
   renvoie l'élément minimal de `T`*, supprime cette valeur de* `T` *et le réordonne de façon à conserver une structure de tas.*

Exercice 6 (Opérations avancées)

Programmer les fonctions suivantes, dans lequelles l'argument `L` *est une liste Python* quelconque *:*

1. `tassifier(L)`
   modifie la liste `L` *de façon à ce qu'elle devienne un tas, contenant le même ensemble d'éléments*
   Pour les courageux : *il est possible de réaliser cette opération en place et en temps linéaire !*
2. `tri_par_tas(L)`
   trie la liste `L` *en deux temps :*
   (a) *transformation en tas*
   (b) *extraction successive des éléments minimaux de façon à reconstituer la liste triée*