TP 7 : Algorithmes (avancés) sur les listes

Exercice 4. Structure de données avancée : le tas, la file à priorité, le tri par tas

Un *tas* est une liste T possédant les propriétés suivantes :

- pour tout élément $T[i]$, on a $T[2i] \leq T[i]$ et $T[2i+1] \leq T[i]$; les éléments $T[2i]$ et $T[2i+1]$ sont appelés les *fils* de $T[i]$, inversement $T[i]$ est appelé le *père* de $T[2i]$ et $T[2i+1]$;
- l'élément $T[1]$ est appelé la *racine* du tas ; il découle des propriétés précédentes que c'est le plus grand élément du tas.

1. Écrivez une procédure `EstTas` qui prend en entrée une liste `L` et renvoie `true` ou `false` selon que `L` est un tas ou pas.

Une *file à priorité* est une structure de données dans laquelle on peut ajouter et retirer des éléments, et dans laquelle l'élément qui sort est toujours le plus grand élément de la file à priorité. On pourrait utiliser une simple liste des élément et chercher le maximum à chaque fois qu'on veut sortir un élément, mais on va améliorer l'efficacité en utilisant un tas.

2. Écrivez une procédure `DescendreTas` qui prend en entrée un (quasi-)tas `T` et un indice `i` tels que l'élément `T[i]` est plus petit que l'un de ses fils, et qui renvoie le tas dans lequel on a descendu l'élément `T[i]` de sorte à avoir un véritable tas.
3. Écrivez une procédure `RemonterTas` qui prend en entrée un (quasi-)tas `T` et un indice `i` tels que l'élément `T[i]` est plus grand que son père, et qui renvoie le tas dans lequel on a remonté l'élément `T[i]` de sorte à avoir un véritable tas.
4. Utilisez les deux procédures précédentes pour écrire en utilisant un tas : `CreeFilePriorite`, `EstVideFilePriorite`, `AjouteFilePriorite`, `SortieFilePriorite`, `RetireFilePriorite` réalisant les mêmes actions que dans l'exercice 2. mais pour une file à priorité.

L'algorithme de *tri par tas* consiste à transformer la liste à trier en tas (en ajoutant successivement les éléments) puis à transformer le tas en liste triée (en retirant à chaque fois le plus grand élément parmi ceux restants).

5. Écrivez une procédure `TriTas` qui prend en argument une liste `L` et qui renvoie la liste triée en ordre croissant en utilisant l'algorithme de tri par tas.

Exercice 5. Évaluation du temps de calcul

1. Reprenez chaque procédure écrite dans les exercices 3. et 4. sauf `TriRapide`, et pour chaque procédure donnez une estimation du nombre d'« opérations élémentaires » réalisées par la procédure en fonction de la taille n de la liste en entrée. On considèrera comme « opération élémentaire » une comparaison, l'accès à un élément d'une liste, une affectation, etc. On comptera « à constante multiplicative près » : le résultat sera parmi $O(1)$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, etc.

Vous avez dû trouver qu'il fallait $O(n \log n)$ opérations pour transformer une liste en tas.

2. On considère la liste `L` et on suppose qu'elle a la structure de tas à partir de l'indice `i` ; montrer comment on peut lui donner la structure de tas à partir de l'indice `i-1`.
3. Écrivez une procédure `Entasse` qui prend en entrée une liste `L` et qui renvoie un tas contenant les éléments de `L` en utilisant le procédé de la question précédente.
4. Montrez que `Entasse` utilise $O(n)$ opérations élémentaires, où n est la taille de la liste en entrée.