Exercice 2 : tri fusion naturel

22 mai

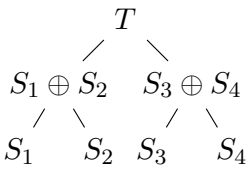
On remarque qu'en pratique les tris par comparaison ont rarement des entrées trop désordonnées. On sait par ailleurs que le tri rapide a de bonnes performances sur des tableaux bien désordonnés ce qui ne correspond donc pas forcément à la réalité. C'est pourquoi, des modifications récentes des librairies utilisent des algorithmes de tri qui sont performants sur des entrées "pseudo-triées".

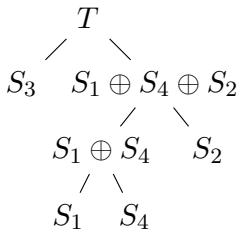
1. Quelle mesure peut permettre de mesurer la "quantité de désordre" d'une liste à trier ?

   Toute liste ℓ non vide peut être décomposée en $\rho \geqslant 1$ séquences croissantes maximales, qui sont des sous listes croissantes $\ell_1, \ldots, \ell_\rho$ dont la concaténation fait ℓ et telles que pour tout $i \in \{1, \ldots, \rho - 1\}$ le dernier élément de ℓ_i est strictement plus grand que le premier élément de ℓ_{i+1}.
   On définit la SCM d'une liste ℓ comme la liste de listes de taille ρ qui contient, dans l'ordre, les séquences définies précédemment.
   Exemple : Si on considère la liste : `[2;4;20;29;3;3;4;1;1;4;32;8;11]` alors sa SCM est représentée par `[[2;4;20;29];[3;3;4];[1;1;4;32];[8;11]]`.
2. Ecrire une fonction `scm : int list->int list list` telle que `scm l` renvoie la SCM de l'entrée `l`. On attend une complexité linéaire en la taille de la liste passée en entrée.
3. Ecrire une fonction `fusion : int list->int list->int list` telle que si on considère deux listes triées `l1,l2`, `fusion l1 l2` renvoie une liste triée qui contient les éléments de `l1` et ceux de `l2`. Quelle est la complexité (nombre de comparaisons) de votre fonction dans le pire cas ?

   L'objectif ici est, à partir de la SCM d'une liste `l`, de trouver une succession de fusions à effectuer sur les séquences la composant pour obtenir une liste triée composée des éléments de `l`. Une représentation d'une succession de fusions peut être faite par un arbre binaire strict dont les feuilles sont les listes et un noeud interne correspond à la fusion des deux listes triées obtenues par la représentation de chacun de ses deux fils. L'arbre choisi aura une influence sur la complexité finale obtenue.

   Exemple : Considérons de nouveau la liste : `[2;4;20;29;3;3;4;1;1;4;32;8;11]` avec sa SCM `[[2;4;20;29];[3;3;4];[1;1;4;32];[8;11]]`. Si on note $S_1 = [2; 4; 20; 29]$, $S2 = [3; 3; 4]$, $S_3 = [1; 1; 4; 32]$ et $S_4 = [8; 11]$ alors on peut obtenir l'un des deux arbres suivants pour représenter la succession de fusions :





4. Dans chacun des deux cas, donner le nombre de comparaisons total de l'ensemble des fusions effectuées sur l'exemple.

   On utilisera le type suivant pour décrire un arbre binaire strict :
   ```
   type arbre = Feuille of list | Noeud of (arbre*arbre)
   ```

   Exemple :
5. Ecrire une fonction `sort_tree : arbre->list` qui à partir de l'arbre des fusions renvoie la liste triée.
6. Une solution est de construire un arbre le plus équilibré possible (arbre binaire strict complet, c'est-à-dire tel que toutes les feuilles sont à même hauteur ou ont une différence de hauteur d'au plus un).
   Ecrire une fonction `constr_arbre : int list list-> arbre` qui construit un tel arbre des fusions. Quelle est la complexité de votre construction ?
7. Ecrire une fonction `tri : int list->int list` qui trie une liste passée en entrée suivant l'algorithme proposé. En déduire une fonction de tri.
8. Déterminer le coût de votre algorithme en fonction de la taille `n` de la liste et de ρ, le nombre de SCM composant la liste.
9. Revenons au cas général où l'arbre des fusions peut-être quelconque : donner une formule qui décrit la complexité de l'algorithme de tri en fonction des profondeurs des différents noeuds de l'arbre et de la taille des séquences de la SCM. A l'aide d'un algorithme au programme, commenter la complexité d'un algorithme glouton qui permettrait de construire un arbre des fusions optimal. Quelle est la complexité de sa construction ?