Structures de données et algorithmes

Feuille de TP n°1

Tris de tableaux de flottants (insertion, sélection, fusion, rapide, Shell)

Exercice 1. Comparaison de tris élémentaires de tableaux

a) Écrivez un programme C qui permette de comparer les exécutions de fonctions de tri élémentaires vues en TD et en cours : tri par insertion (fonctions *insertion_float* et *insertion* de la feuille de TD n°1), et tri par sélection (cours n°1).

Pour ce faire, votre programme doit

- recevoir comme arguments en ligne de commande la longueur du tableau à trier et le mode d'initialisation choisie pour celui-ci (génération « aléatoire » ou saisie au clavier) ;
- réserver de la mémoire sur le tas pour initialiser autant de tableaux de flottants identiques que de fonctions de tri à tester ;
- initialiser et afficher le tableau initial ;
- pour chacune des fonctions de tri à tester, exécuter la fonction sur une des occurrences du tableau initial, afficher le résultat (si le tableau n'est pas trop long), ainsi que le nombre de comparaisons de valeurs et d'affectations de valeurs effectuées au cours de cette exécution ;
- libérer la mémoire réservée sur le tas.

N.B. Pour calculer et afficher le nombre de comparaisons et d'affectations de valeurs requises par l'exécution des fonctions de tri, il vous faudra modifier légèrement le code de celles-ci, tel que vous l'avez vu en cours ou en TD.

b) Lancer plusieurs exécutions de votre programme, pour des longueurs de tableaux de plus en plus grandes et vérifier que les affichages obtenus correspondent aux calculs de coût faits en cours et en TD.

Exercice 2 (bonus). Tri Shell

Le tri par insertion dans sa forme standard (fonction *insertion* de la feuille de TD n°1) ne compare et n'échange que des valeurs contiguës. Il doit donc effectuer un nombre de comparaisons et d'échanges au moins égal au nombre d'inversions de la suite de valeurs à trier (cf. les exercices 2 et 3 de la feuille de TD n°1). Si les n valeurs à trier sont initialement dans la configuration

2	1	1	$\cdots$	1	1

il faudra $n-1$ échanges, alors qu'un seul suffirait.

Pour remédier à cette « faiblesse », le tri Shell commence par comparer, pour les ordonner, des paires de valeurs éloignées l'une de l'autre puis recommence en comparant des paires de valeurs plus proches l'une de l'autre, etc. La fonction C *shell* ci-dessous implémente cette stratégie, dans laquelle la « distance [1] » entre deux valeurs comparées (et éventuellement échangées) vaut successivement $\cdots$ 9841 3280 1093 364 121 40 13 $\cdots$

1. Deux valeurs contiguës sont à distance 1 l'une de l'autre.

```
1  void shell(item *tab, int g, int d) {
2    int i, j, h;
3    for (h = 1; 3*h <= d-g-1; h = 3*h + 1);
4    for ( ; h > 0; h /= 3) {
5      for (i = g+h; i <= d; ++i) {
6        for (j = i; j >= g+h && less(tab[j], tab[j-h]); j -= h) {
7          exch(tab[j-h], tab[j]);
8        }
9      }
10   }
11 }
```

a) Expliquez la boucle de la ligne 3.

Démontrez que la fonction *shell* trie les valeurs du tableau tab comprises entre les indices g et d.

b) On s'intéresse de plus près au triplet de boucles *for* imbriquées : la boucle externe (indice h, ligne 4), la boucle médiane (indice i, ligne 5) et la boucle interne (indice j, ligne 6). On note $n = d - g + 1$ la longueur du segment de tableau à trier.

On dit qu'une suite de valeurs $v_0, v_1, v_2, \ldots$ est h-triée si $v_p \leqslant v_{p+h} \leqslant v_{p+2h} \leqslant \cdots$ pour $p = 0, \ldots, h-1$.

Après une itération de la boucle externe, que pouvez-vous dire de la suite $tab[g], tab[g+1], \ldots, tab[d]$?

Soit k_1 et k_2, $k_2 < k_1$, deux valeurs d'indice correspondant à deux itérations de la boucle externe ; que pensez-vous pouvoir dire de la suite $tab[g], tab[g+1], \ldots, tab[d]$ après l'itération d'indice k_2 ?

c) [Lisez l'énoncé de cette question mais ne cherchez pas à faire la démonstration pendant la séance de TP] Démontrez que le coût d'une exécution de la fonction *shell* est $O(n^{3/2})$ dans le cas le plus défavorable.

d) Enrichissez le programme des exercices 1 et 2 pour y inclure de quoi comparer et tester le tri Shell. Vérifiez que les nombres de comparaisons et d'affectations de flottants affichés sont cohérents avec les informations données ci-dessus.

Exercice 3 (bonus). Tri-fusion et tri rapide

Enrichissez le programme de l'exercice 1 pour y inclure de quoi comparer et tester les deux tris intrinsèquement « récursifs » vus en cours, à savoir le tri-fusion et le tri rapide.