Algorithmique

Laurent Debize

TIIS2

Tri fusion

L'algorithme de tri par fusion est construit suivant le paradigme « diviser pour régner » :

1. Il divise la séquence de n nombres à trier en deux sous-séquences de taille $n/2$.
2. Il trie récursivement les deux sous-séquences.
3. Il fusionne les deux sous-séquences triées pour produire la séquence complète triée.

La récursion termine quand la sous-séquence à trier est de longueur 1... car une telle séquence est toujours triée.

Tri fusion

La fusion

Le principe de cette fusion est simple : à chaque étape, on compare les éléments minimaux des deux sous-listes triées, le plus petit des deux étant l'élément minimal de l'ensemble on le met de côté et on recommence. On conçoit ainsi un algorithme FUSIONNER qui prend en entrée un tableau A et trois entiers, p, q et r, tels que $p \leq q < r$ et tels que les tableaux $A[p..q]$ et $A[q+1..r]$ soient triés.

Tri fusion

```
FUSIONNER(A, p, q, r)
  i ← p                                   indice servant à parcourir le tableau A[p..q]
  j ← q + 1                               indice servant à parcourir le tableau A[q + 1..r]
  Soit C un tableau de taille r − p + 1   tableau temporaire dans lequel on construit le résultat
  k ← 1                                   indice servant à parcourir le tableau temporaire
  tant que i ≤ q et j ≤ r faire           boucle de fusion
    si A[i] < A[j] alors C[k] ← A[i]
                         i ← i + 1
                   sinon C[k] ← A[j]
                         j ← j + 1
    k ← k + 1
  tant que i ≤ q faire C[k] ← A[i]        on incorpore dans C les éléments de A[p..q]
                       i ← i + 1            qui n'y seraient pas encore ; s'il y en a,
                       k ← k + 1            les éléments de A[q + 1..r] sont déjà tous dans C
  tant que j ≤ r faire C[k] ← A[j]        on incorpore dans C les éléments de A[q + 1..r]
                       j ← j + 1            qui n'y seraient pas encore ; s'il y en a,
                       k ← k + 1            les éléments de A[p..q] sont déjà tous dans C
  pour k ← 1 à r − p + 1 faire            on recopie le résultat dans le tableau originel
      A[p + k − 1] ← C[k]
```

```
TRI-FUSION(A, p, r)
  si p < r alors q ← ⌊(p + r)/2⌋
                 TRI-FUSION(A, p, q)
                 TRI-FUSION(A, q + 1, r)
                 FUSIONNER(A, p, q, r)
```

Exercice 1

Vous répondrez à toutes les questions dans un document Word que vous m'enverrez par mail ensuite en précisant dans l'objet : **TIIS2**.

1. Implémenter cet algorithme et le tester sur un tableau quelconque.
2. On se propose d'estimer la complexité de cet algorithme. Pour cela, on va compter le nombre d'itérations.
   - Placer un compteur à l'intérieur de chaque boucle.
   - Afficher la valeur de ce compteur à la fin de l'algorithme.
   - Faire varier la taille n du tableau entre 1 et 10 (on pourra se servir d'une boucle qui entoure tout l'algorithme)
   - Relever la valeur du compteur pour chaque taille n du tableau. Placer ces valeurs dans un tableau Excel.
   - Tracer sur un graphique Excel la valeur du compteur en fonction de la taille du tableau
   - Faire un clic droit sur la courbe, et ajouter une courbe de tendance. Dans les options, sélectionner **Puissance**, et cocher **Afficher l'équation sur le graphique**.
   - Quelle est l'équation qui s'affiche sur le graphique ? Cette équation traduit la complexité de l'algorithme en fonction de la taille des données, ici n.