Informatique – TP n° 3 – Quelques algorithmes de tri

Nous avons déjà vu comment trier un tableau par insertion (voir TP n° 1). Voici trois autres méthodes pour trier un tableau.

Exercice 1 – Le tri à bulles

On considère un tableau T constitué de n cases. Le tri à bulles consiste à parcourir l'ensemble des indices $i \in [\![1, n-1]\!]$, du tableau, et, à chaque étape, à comparer $T[i]$ et $T[i+1]$: s'ils sont dans le bon ordre, on ne fait rien, sinon on les échange, puis on passe à l'indice suivant.

1. (a) Effectuer étape par étape (à la main) cet algorithme sur le tableau $(5, 1, 2, 3, 4)$.
   (b) Même question sur le tableau $(5, 4, 3, 2, 1)$.
   (c) Après un parcours du tableau, le tableau obtenu est-il trié ?
   (d) Montrer qu'après avoir parcouru le tableau, l'élément maximal se retrouve en bout de tableau.
2. (a) Justifier qu'en parcourant $n-1$ fois le tableau de la façon précédente, le tableau obtenu est trié. Justifier que si n est la taille du tableau, lors du j-ième parcours, on peut se contenter d'aller seulement jusqu'à l'indice $n-j$.
   (b) Écrire une procédure prenant en paramètre un tableau de taille n non trié, effectuant sur ce tableau les opérations décrites ci-dessus, et renvoyant le tableau trié.
3. On note C_n le nombre d'opérations effectuées lors d'une exécution de cet algorithme (affectation ou test de comparaison)
   (a) Donner un encadrement de C_n.
   (b) Justifier que $C_n = O(n^2)$.
   (c) Justifier qu'il n'existe pas de réel $\alpha < 2$ tel que $C_n = O(n^\alpha)$.
4. Proposer des améliorations à cet algorithme.

Exercice 2 – Le tri rapide

Le tri rapide est un algorithme récursif. Soit T un tableau de taille n.
Si le tableau est de taille 0 ou de taille 1, on ne fait rien
On commence par choisir dans un premier temps un élément $x = T[i]$ au hasard dans le tableau. Dans un autre tableau de même taille, on regroupe en début de tableau toutes les valeurs du tableau (dans un ordre quelconque) strictement plus petites que x, et on regroupe en fin de tableau toutes les valeurs strictement plus grandes que x. Ces valeurs sont séparées en milieu de tableau par la valeur x, éventuellement répétée autant de fois que x apparaît dans le tableau initial.
On applique ensuite récursivement l'algorithme de tri au sous-tableau constitué des éléments strictement plus petits que x, et au sous-tableau des éléments strictement plus grands que x.
Pour pouvoir appliquer l'algorithme de tri à un sous-tableau contitué des éléments d'indice a à b, on passera en paramètre la tableau entier, ainsi, que l'indice initial a et l'indice final b de l'ensemble des éléments à trier. Ainsi, les conditions initiales (tableaux de taille 0 ou 1) sont traduites par l'égalité $b = a$ (taille 1) ou l'inégalité $b < a$ (taille 0).

1. Justifier que quel que soit le tableau initial T, l'algorithme s'arrête et renvoie le tableau trié. On précisera soigneusement l'opportunité de la condition d'arrêt donnée pour les tableaux de taille 0.
2. Écrire en Pascal une procédure utilisant cet algorithme pour trier un tableau passé en paramètre.
3. On note C_n le nombre d'opérations effectuées pour trié un tableau T donné par cet algorithme.
4. On suppose que les entrées du tableau sont deux à deux distinctes.
   (a) On suppose qu'à chaque appel récursif de la procédure, le choix aléatoire de x donne le plus petit élément du tableau. Soit M_n le nombre d'opérations nécessaire (affectation ou comparaison) pour trier le tableau dans ce cas. Montrer qu'il existe $\alpha > 0$ tel que $M_n = \alpha n(n+1)$.

(b) Soit C_n le nombre d'opérations effectuées pour trier le tableau T. Montrer par récurrence forte sur n que $C_n \leqslant M_n$. En déduire que $C_n = O(n^2)$.

5. On suppose toujours que les entrées du tableau sont deux à deux distinctes, et on suppose qu'il existe k tel que $n = 2^k - 1$.

   (a) On suppose que dans la première étape, on choisit x égal à la médiane des éléments du tableau. Justifier qu'on est ramené au tri de deux tableaux de taille $2^{k-1} - 1$.

   (b) On suppose qu'à chaque appel récursif, le réel x est la médiane des éléments à trier. Calculer dans ce cas le nombre d'opérations effectuées pour trier le tableau.

   On peut montrer que ceci est le meilleur des cas.

Exercice 3 – Tri-fusion

Il s'agit aussi d'un algorithme récursif. Cette fois, on commence par couper le tableau T de taille n en 2 parts égales (ou presque) :

- si n est pair, $n = 2k$, on coupe en deux tableaux de taille k, l'un contitué des indices de 1 à k, l'autre des indices de $k+1$ à n ;
- si n est impair, $n = 2k+1$, on considère un premier tableau de taille k constitué des indices de 1 à k, et un deuxième tableau de taille $k+1$, constitué des indices de k à n.

On trie chacun de ces deux tableaux, grâce à un appel récursif de la procédure. On obtient donc deux tableaux triés, qu'il faut ensuite fusionner.

1. Expliquer comment faire la fusion de deux tableaux triés en un seul tableau trié, en parcourant simultanément les deux tableaux.

2. Écrire en Pascal une procédure triant un tableau à l'aide de cet algorithme. On utilisera un tableau auxiliaire déclaré globalement pour faire la fusion.

3. On suppose qu'il existe k tel que $n = 2^k$. On note C_n le nombre d'affectations ou de tests effectués lors du tri d'un tableau T donné. Justifier qu'il existe des réels strictement positifs α et β tels que $\alpha < \beta$ et :

$$C'_{2^{k-1}} + C''_{2^{k-1}} + \alpha 2^k \leqslant C_{2^k} = C_n \leqslant C'_{2^{k-1}} + C''_{2^{k-1}} + \beta 2^k,$$

   où $C'_{2^{k-1}}$, $C''_{2^{k-1}}$, sont les nombres d'opérations effectuées respectivement lors du tri de deux tableaux de taille 2^{k-1}.

   En déduire que :

$$\alpha(k+1)n \leqslant C_n \leqslant \beta(k+1)n.$$

   En déduire que $C_n = O(n \ln n)$, et qu'on ne peut pas trouver $(a, b) \in \mathbb{R}^2$ tel que $a < 1$ ou tel que $a = 1$ et $b < 1$, vérifiant :

$$C_n = O(n^a \ln^b n).$$