Algorithmique et programmation
Travaux dirigés, 16 et 19 novembre 1999

Louis Granboulan et Laurent Mauborgne

1 Tri Shell

1. Rappels sur le tri par insertion directe

   Le tri par insertion dierecte consiste à insérer un par un les éléments de la liste dans le tableau des éléments déjà triés.

   (a) Pour une permutation σ, on appelle inversion un couple $i < j$ tel que $\sigma(i) > \sigma(j)$. Évaluer le nombre moyen d'inversions d'une permutation de n éléments.

   (b) Étudier la complexité de l'algorithme en nombre de comparaisons et d'affectations.

   (c) Réfléchir aux différentes façons d'améliorer cet algorithme

2. Tri Shell

   L'idée est d'essayer de gagner sur le nombre d'affectations en insérant dans des sous-listes (et donc en faisant faire de plus grands déplacements). L'algorithme consiste en un tri par insertion dans des sous-listes contenant de plus en plus d'éléments, de telle sorte que les éléments soient de plus en plus proches de leur position finale (D. Shell, 1959).
   On dit qu'un tableau est h-trié si les h sous-tableaux regroupant les éléments d'indice différant d'un multiple de h sont triés. C'est-à-dire que les suites $(x_i, x_{i+h}, x_{i+2h}, ...)$ pour $i = 1, ..., h$ sont triées.
   Pour le tri Shell, on choisit une suite entière décroissante $h_t, h_{t-1}, ..., h_1$ avec $h_1 = 1$. On commence par h_t-trier le tableau, puis on continuer avec h_{t-1}, etc. Le h_s-tri se fait par exemple par insertion directe, pour chaque sous-tableau.

   (a) Prouver la correction de cet algorithme de tri.

   (b) Prouver que le h-tri d'un ensemble k-trié donne un ensemble k-trié. En déduire que le cas $h_s = 2^{s-1}$ n'est pas optimal.

   (c) Évaluer le nombre moyen d'inversions d'une permutation 2-triée.

      i. Calculer le nombre de permutations 2-triées.

      ii. Montrer que la somme des nombres d'inversions des permutations 2-triées de n éléments (avec $k = \lfloor n/2 \rfloor$) est $A_n = \sum_{i=0..k, j=0..k} |i-j| \binom{i+j}{j} \binom{n-i-j-1}{k-j}$.

      iii. Prouver $A_{2k+1} = 2A_{2k}$. On peut montrer que $A_{2k} = k4^{k-1}$.

      En déduire la complexité de l'algorithme dans le cas où on fait un 2-tri suivi d'un 1-tri.

   (d) Regarder le cas où on fait un h-tri suivi d'un 1-tri.

   (e) Regarder le cas où $h_s = 2^s - 1$.

Knuth, *The Art of Computer Programming, vol. 3 :* ***Sorting and Searching***, pp. 87–92.

2 Chaînes d'additions

Le but est de calculer de manière aussi efficace que possible la puissance entière d'un nombre.

1. Trouver différentes façons de calculer la puissance entière d'un nombre en minimisant le nombre de multiplications.

2. Les chaînes d'additions

   Une chaîne d'additions pour n est une suite $a_0, \ldots, a_r$ telle que $a_0 = 1$, $a_r = n$ et tout élément de la liste est la somme de deux éléments précédents ($a_i = a_j + a_k$ et $j, k < i$).

   (a) On appelle $l(n)$ la longueur de la plus petite chaîne d'additions pour n. Déduire des méthodes précédentes des propriétés de $l(n)$.

   (b) On appelle doublage un élément d'une chaîne d'additions un élément $a_i = a_{i-1} + a_{i-1}$. Soit d le nombre de doublages d'une chaîne d'additions pour n. Montrer que $n \leqslant 2^{d-1} F_{r-d+3}$ où (F_k) est la suite de Fibonacci.

   (c) Trouver $l(n)$ si $n = 2^A$, $n = 2^A + 2^B$, $n = 2^A + 2^B + 2^C$.

Knuth, *The Art of Computer Programming, vol. 2 :* ***Seminumerical Algorithms***, pp. 398–422.