Algorithmique
TD 5. Semaine du 5/11/2001

Tri par insertion

On considère la méthode suivante, réalisant le tri par insertion d'un tableau T à N éléments, numérotés de 1 à N :

- pour chaque $i \in [2, \ldots, N]$,
  - $j := i - 1$.
  - tant que $(j > 0)$ et $T[j] > T[j+1]$,
    - échanger les contenus de $T[j]$ et $T[j+1]$.
    - $j := j - 1$.

Rappelons que dans le pire des cas, le nombre d'échanges nécessaires est égal à $1 + 2 + \ldots (N-1) = \frac{N(N-1)}{2}$.

Exercice 1

Soient $(a_1, \ldots, a_p)$ et $(b_1, \ldots, b_p)$ deux suites d'entiers dont les éléments sont triés par ordre croissant. Soit $T = (a_1, b_1, a_2, b_2, \ldots, a_p, b_p)$.

Démontrer, par récurrence sur p, que le tri par insertion de T nécessite, dans le pire des cas, seulement $\frac{p(p+1)}{2}$ échanges. On notera que lorsque i prend pour la première fois la valeur $2p-1$, c'est-à-dire lorsque l'on atteint pour la première fois a_p,

- T est trié jusqu'à la case $2p-2$,
- T est de l'une des formes suivantes :
  - $(\ldots, a_{p-1}, b_{p-1-k}, \ldots, b_{p-1}, a_p, b_p)$ pour un certain $k \geq 0$, ou bien,
  - $(\ldots, b_{p-1}, a_{p-1-k}, \ldots, a_{p-1}, a_p, b_p)$ pour un certain $k \geq 0$.

Exercice 2

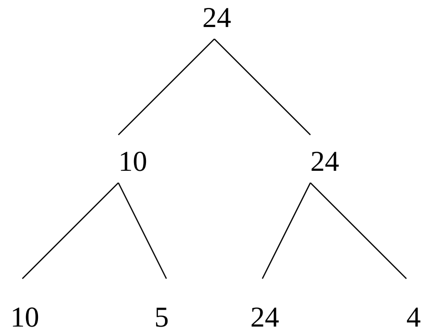
On considère à présent un tableau T arbitraire de taille $n = 2 \times p$, auquel on applique la méthode suivante :

- trier sur place, par insertion, le sous-tableau de T d'indices paires,
- trier sur place, par insertion, le sous-tableau de T d'indices impaires,
- trier, par insertion, le tableau T résultant.

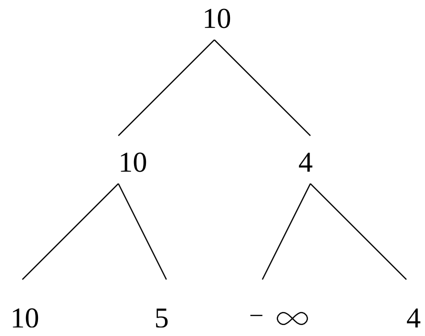
Quel est, dans le pire des cas, le coût de ce tri ? Comparer la valeur obtenue avec celle du tri par insertion simple. Est-elle meilleure ou pire ?

Tri par sélection avec arbre

On considère une variante du tri par sélection qui utilise des arbres. Par exemple, le tableau `[10 5 24 4]` sera trié comme suit : on construit un arbre binaire équilibré dont les feuilles sont les éléments du tableau, et où chaque autre noeud est le maximum de ses fils.



On prend le maximum (la racine) qu'on place, à sa position correcte (*i.e.* en dernière position) dans un tableau auxiliaire. On remplace dans l'arbre cette valeur par $-\infty$, et on recalcule les noeuds qui changent de valeur :



On recommence jusqu'à ce que tous les éléments soient enlevés.

Exercice 3

- Appliquer cet algorithme au tableau `[10 8 29 3 49 3 22 10]`
- Combien de fois compare-t-on $-\infty$ avec $-\infty$, en triant un tableau de 2^n éléments ?
- Analyser la complexité de cet algorithme dans le pire des cas (nombre de comparaisons).