903. Exemples d'algorithmes de tri. Complexité.

Définition 1. Un algorithme de tri prend un tableau d'éléments $t_1, \dots, t_n$ d'un ensemble ordonné $(X, \leqslant)$ et détermine implicitement une permutation σ de $\{1, \dots, n\}$ telle que $t_{\sigma(1)} \leqslant \cdots \leqslant t_{\sigma(n)}$.

Remarque 2. On mesure la complexité d'un algorithme de tri par le nombre de comparaisons (parfois d'échanges) qu'il effectue.

1 Algorithmes naïfs

Tri par sélection

procédure SÉLECTION(t, n)
  pour $k = n, \dots, 1$ **faire** ▷ Invariant : « les $n - k$ derniers sont bien placés »
    $m \leftarrow 1$
    pour $i = 2, \dots, k$ **faire**
      si $t_i > t_m$ **alors**
        $m \leftarrow i$
    ÉCHANGER(m, k)
  renvoyer t

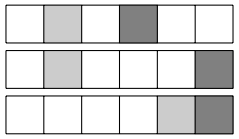

FIGURE 1 – Les premières étapes d'un tri par sélection.

Remarque 3. Le tri par sélection effectue $O(n^2)$ comparaisons et $O(n)$ échanges.

Tri par insertion (non stable, me semble-t-il)

procédure INSERTION(t, n)
  pour $k = 1, \dots, n$ **faire** ▷ Invariant : « les $k - 1$ premiers sont bien placés »
    $i \leftarrow$ RECHERCHERPOSITION(t_k dans $(t_1, \dots, t_{k-1})$ trié)
    INSÉRER(t_k dans $(t_1, \dots, t_{k-1})$ à la position i)
  renvoyer t

Remarque 4. Si la recherche est effectuée linéairement, le tri par insertion effectue $O(n^2)$ comparaisons dans le pire cas et $O(n)$ comparaisons dans le meilleur cas.

Remarque 5. Si la recherche est effectuée par dichotomie, on obtient un nombre logarithmique de comparaisons mais l'insertion reste linéaire en nombre d'échanges. Cette version du tri par insertion effectue donc $O(n \log n)$ comparaisons dans tous les cas et $O(n^2)$ échanges dans le pire cas.

Remarque 6. Si on remplace t par une liste chaînée, on peut effectuer l'insertion en une étape mais la recherche sera forcément en nombre linéaire de comparaisons. Dans le cas d'une liste triée, $O(n)$ comparaisons sont effectuées.

Tri à bulles

procédure BULLES(t, n)
  pour $k = n, \dots, 1$ **faire** ▷ Invariant : « les $n - k$ derniers sont bien placés »
    pour $i = 1, \dots, k - 1$ **faire**
      si $t_i > t_{i+1}$ **alors**
        ÉCHANGER($i, i + 1$)
  renvoyer t

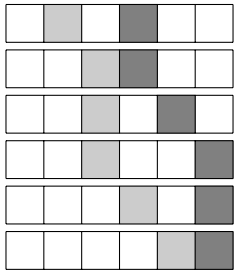

FIGURE 2 – Les premières étapes d'un tri à bulles.

Proposition 7. Le tri à bulles effectue $O(n^2)$ comparaisons et le nombre minimum de transpositions successives pour trier une séquence, à savoir le nombre d'inversions de σ où σ est la permutation de tri.

2 Algorithmes optimaux

Théorème 8. Tout tri par comparaisons effectue dans le pire cas $\Omega(n \log n)$ comparaisons.

Tri rapide (non stable)

procédure RAPIDE(t, d, f) ▷ tableau, début et fin
  si $d < f$ **alors**
    $i \leftarrow$ CHOISIRPIVOT(t)
    $p \leftarrow t_i$
    ÉCHANGER(i, f)
    $\ell \leftarrow d$
    pour $i = d, \ldots, f-1$ **faire** ▷ Invariant : « $\{t_d, \ldots, t_{\ell-1}\} < p \leqslant \{t_\ell, \ldots, t_{i-1}\}$ »
      si $t_i < p$ **alors**
        ÉCHANGER(i, ℓ)
        $\ell \leftarrow \ell + 1$
    ÉCHANGER(f, ℓ)
    RAPIDE($t, d, \ell - 1$)
    RAPIDE($t, \ell + 1, f$)

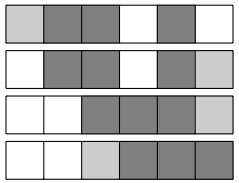

FIGURE 3 – Partition d'un tri rapide. Le pivot est le premier élément, placé en fin. Ensuite, les éléments plus petits que le pivot sont regroupés à gauche du tableau, et enfin le pivot est replacé.

Tri fusion

procédure FUSION($t_1, \ldots, t_n$)
  si $n > 1$ **alors**
    $u \leftarrow$ FUSION($t_1, \ldots, t_{\lfloor n/2 \rfloor}$)
    $v \leftarrow$ FUSION($t_{\lfloor n/2 \rfloor + 1}, \ldots, t_n$)
    $a, b \leftarrow 1, 1$
    pour $i = 1, \ldots, n$ **faire** ▷ Invariant : « les $i-1$ premiers sont bien placés »
      si $b > |v|$ ou $u_a \leqslant v_b$ **alors** ▷ C'est donc un tri stable
        $t_i \leftarrow u_a$
        $a \leftarrow a + 1$
      sinon
        $t_i \leftarrow v_b$
        $b \leftarrow b + 1$
    renvoyer t

Pivot	Cas moyen	Pire cas
le premier élément	$O(n \log n)$	$O(n^2)$
un élément au hasard	$O(n \log n)$	$O(n^2)$
l'élément médian	$O(n \log n)$	$O(n \log n)$

TABLE 1 – Nombre de comparaisons pour différents choix de pivot.

Développement 1. Le tri fusion effectue toujours $O(n \log n)$ comparaisons, tandis que le tri rapide choisissant le premier élément comme pivot effectue $O(n \log n)$ comparaisons en moyenne, $O(n^2)$ dans le pire cas, celui d'un tableau trié.

Définition 9. Un tas est un arbre (binaire) tournoi quasi parfait, c'est-à-dire dont :

1. chaque nœud interne a une clé plus grande que celles de ses fils ;
2. toutes les feuilles sont sur au plus deux niveaux, le moins profond étant complet et les feuilles du plus profond niveau le plus regroupées à gauche possible.

Tri par tas

procédure TAS($t_1, \ldots, t_n$)
  pour $i = \lfloor n/2 \rfloor, \ldots, 1$ **faire** ▷ Complexité linéaire
    TAMISERVERSBAS(i, t, n)
  pour $f = n, \ldots, 1$ **faire** ▷ Invariant : « les $n - f$ derniers sont bien placés »
    ÉCHANGER($1, f$)
    TAMISERVERSBAS($1, t, f - 1$)
  renvoyer t
procédure TAMISERVERSBAS(i, t, f)
  si $2i + 1 \leqslant f$ et $t_{2i+1} > t_{2i}$ et $t_{2i+1} > t_i$ **alors**
    ÉCHANGER($i, 2i + 1$)
    TAMISERVERSBAS($2i + 1, t, f$)
  sinon si $2i \leqslant f$ et $t_{2i} > t_i$ **alors**
    ÉCHANGER($i, 2i$)
    TAMISERVERSBAS($2i, t, f$)

Développement 2. Le tri par tas est bien un tri qui effectue toujours $O(n \log n)$ comparaisons.

Références

BBC
Cormen