TRIS

Soit E un ensemble d'objets. On considère une application v, appelée critère de tri, de E dans un ensemble O totalement ordonné. Par l'intermédiaire de cette application v, on peut comparer les éléments de E : soit x, y deux éléments de E, on dit que x est plus petit (resp. plus grand) que y si et seulement si $v(x) \leqslant v(y)$ (resp. $v(x) \geqslant v(y)$). Deux éléments x, y tels que $v(x) = v(y)$ sont dit indiscernables.

À toute suite finie s de E, de longueur n, on peut associer l'ensemble des permutations de s, c'est-à-dire les suites finies contenant les n éléments de s dans un certain ordre : il existe $n!$ permutations de s. Parmi ces permutations $(a_1, \ldots, a_n)$, certaines sont ordonnées

- par ordre croissant : pour tout $(i, j) \in \{1, \ldots, n\}$, si $i \leqslant j$, alors $v(a_i) \leqslant v(a_j)$
- par ordre décroissant : pour tout $(i, j) \in \{1, \ldots, n\}$, si $i \leqslant j$, alors $v(a_i) \geqslant v(a_j)$

Trier s, c'est construire une permutation croissante de s. Une fonction de tri est alors une fonction qui prend en paramètre une quelconque suite s et qui retourne une permutation de s ordonnée par ordre croissant. Elle est dite générique si elle n'utilise pas d'autre information que le critère de tri pour trier la suite.

Théorème 1 *Un algorithme générique de tri a une complexité pire-cas au moins de l'ordre de* $n \log n$.
Si $c(n)$ *est la complexité d'un algorithme de tri générique, alors* $n \log n = O(\ c(n)\ )$.

Ce théorème n'interdit pas :

- que la complexité meilleur-cas soit meilleure ;
- que de algorithmes de tri non génériques puissent être plus rapides (tri radix, par exemple).

Dans la suite, la concaténation de deux suites finies s et t est notée $s.t$; si f est une suite finie et $x = f_k$ un élément de cette suite, $f \setminus x$ signifie la suite f privé de son k-ème élément.

1 Tris quadratiques

1.1 Tri par sélection

Principe récursif : f une suite finie

- on cherche parmi les éléments de f un plus petit élément x (*i.e.* tel que $v(x)$ soit minimal) ;
- on trie récursivement la suite $g = f \setminus x$, on obtient $s = (b_1, \ldots, b_{n-1})$;
- alors $x.s = (x, b_1, \ldots, b_n)$ est un tri de f.

le cas de base étant l'ensemble vide, à qui on associe la suite vide.

Principe itératif :

- on pose $f_0 = f$, $s_0 = ()$;
- puis pour i variant de 1 à n,
  - on choisit un plus petit élément dans f_i, qu'on note x ;
  - on pose $s_i = s_{i-1}.x$ et $f_i = f_{i-1} \setminus x$;
- alors en fin de boucle, s_n est un tri de F.

1.2 Tri à bulles

Principe itératif (tri sur place) :

- on pose $s_{0,0}$ une permutation de F ;
- pour i variant de 1 à $n - 1$,
  - pour j variant de i à $n - 1$,
    - si le j-ème élément de $s_{i,j}$ (noté x) est plus grand que le $(j + 1)$-ème (noté y), alors on appelle $s_{i,j+1}$ la permutation obtenue en échangeant les deux éléments x et y, sinon $s_{i,j+1} = s_{i,j}$
  - on pose $s_{i+1,0} = s_{i,n}$
- alors en fin de boucle, $s_{n,n}$ est un tri de F.

1.3 Tri par insertion

Principe récursif :
- on choisit parmi les éléments de f un élément x ;
- on trie récursivement l'ensemble $g = f \setminus x$, on obtient $s = (b_1, \ldots, b_{n-1})$;
- puis on insère x en bonne position dans s : on cherche un indice i tel que $b_i \leqslant x \leqslant b_{i+1}$
- alors $(b_1, \ldots, b_i, x, b_{i+1}, \ldots, b_n)$ est un tri de f.

le cas de base étant l'ensemble vide, à qui on associe la suite vide.

Principe itératif :
- on pose $f_0 = f$, $s_0 = ()$;
- puis pour i variant de 1 à n,
  - on choisit un élément dans f_i, qu'on note x ;
  - on insère x en bonne position dans s_{i-1} : on obtient une suite finie s_i, puis on pose $f_i = f_{i-1} \setminus x$;
- alors en fin de boucle, s_n est un tri de f.

2 Tris quasi-linéaires

2.1 Tri fusion

Principe récursif :
- on partitionne f en deux suites g, h de cardinaux $n/2$, $(n+1)/2$;
- on trie récursivement g et h, on obtient deux suites triées t et u ;
- on fusionne les deux suites triées t et u en une seule suite triée s ;
- alors s est un tri de f.

le cas de base étant l'ensemble vide ou un singleton, à qui on associe l'unique permutation possible.

2.2 Tri rapide

Principe récursif :
- on choisit un élément x de f ;
- on partitionne $f \setminus x$ en deux suites g, qui ne contient que des éléments de $f \setminus x$ plus petits que x, et h qui ne contient que des éléments de $f \setminus x$ plus grands que x.
- on trie récursivement g et h, on obtient deux suites triées t et u ;
- on concatène : $s = t.x.u$
- alors s est un tri de f.

le cas de base étant l'ensemble vide ou un singleton, à qui on associe l'unique permutation possible.

3 En pratique

Les suites finies sont représentées par des tableaux ou des listes : ce sont des structures dites « linéaires ».

Une fonction de tri est dite « tri sur place » si et seulement si elle prend une structure en paramètre et elle la modifie de sorte qu'elle soit triée. Avec CAML, on ne peut pratiquer de tri sur place que dans les tableaux, qui sont des structures mutables (avec d'autres langages, on peut disposer de listes mutables ; en CAML, il faut les définir soi-même). L'avantage d'un tri sur place est qu'il ne gaspille pas la mémoire, on peut trier sans espace supplémentaire, hormis quelques case pour ranger des résultats intermédiaires. L'inconvénient est bien sûr que la structure initiale a été modifiée, donc est perdue. Que l'on pratique du tri classique ou sur place, cela ne change rien à l'ordre de grandeur de la complexité.

Une fonction de tri est dite « stable » si deux éléments indiscernables de la structure initiale sont rangés dans le même ordre dans la structure triée.