Option Informatique

TD $n°2$: Tris (1)

Dans cette feuille on s'intéresse aux algorithmes de tri basés sur le principe de la comparaison des éléments [1], et plus précisément à deux tris très rudimentaires : le *tri par sélection* et le *tri par insertion*.

On peut être amené à trier diverses structures de données. Ici on s'intéresse uniquement au tri des *tableaux* ou au tri des *listes*. Une différence fondamentale entre ces structures de données en OCaml [2] est que les listes sont des structures *immuables* alors que les tableaux sont des structures *modifiables* : on ne peut pas modifier une liste, on peut seulement créer une nouvelle liste à partir d'une liste donnée ; par contre, on peut modifier les éléments d'un tableau, de la même façon qu'on peut modifier une référence. Par conséquent, on considèrera :

- qu'un algorithme de tri d'une liste est un algorithme qui retourne une nouvelle liste comportant les mêmes éléments que la première, mais triée ;
- alors qu'un algorithme de tri d'un tableau ne retourne rien mais a pour effet de bord de modifier le tableau donné en argument en le triant.

L'unité de mesure de toutes les complexités calculées sera le nombre de comparaisons d'éléments effectuées. Bien sûr, tous les algorithmes de tris nécéssitent d'autres types d'opérations (accès à un élément d'un tableau ou écriture dans un tableau dans le cas du tri d'un tableau ; accès à l'élément de tête d'une liste ou cons d'un élément et d'une liste dans le cas du tri d'une liste... pour ne citer que ceux-là), mais on ne s'intéressera ici qu'aux comparaisons.

Tri par sélection

Exercice 1.

On explique le principe du tri par sélection sur les tableaux. Pour trier un tableau t de longueur n :

- On détermine l'élément maximal parmi t.(0), ..., t.(n-1) ;
- On permute cet élément avec t.(n-1) ;
- On recommence sur t.(0), ..., t.(n-2) ; etc.

1. Implémenter cet algorithme en OCaml. Écrire des fonctions auxiliaires semble s'imposer.
2. Montrer que dans le pire des cas, le nombre de comparaison effectuées est un $O(n^2)$.
3. Qu'en est-il dans le meilleur des cas ?

Exercice 2.

1. Proposer une variante sur les listes.
2. Implémenter la variante en OCaml. On évitera le recours à la concaténation @ qui est une opération couteuse.

TSVP

1. Il existe d'autres façon d'obtenir des algorithmes de tri, bien que non intuitives.
2. Parmi bien d'autres !

Tri par insertion

Exercice 3.

On explique le principe du tri par insertion sur les listes. Pour trier une liste ℓ :

- On considère deux listes : la liste ℓ_1 des éléments qui sont encore à tirer, initialement ℓ, et la liste ℓ_2 des élément déjà triés, initialement [].
- On prend la tête de ℓ_1 et on l'insère à sa place dans ℓ_2 ;
- On recommence jusqu'à que ce ℓ_1 soit vide.

Bien sûr, dans la description précédente, il n'y a pas de réelle modification des deux listes, simplement des appels récursifs sur des listes modifiées.

1. Implémenter cet algorithme en OCaml. Écrire des fonctions auxiliaires semble s'imposer.
2. Montrer que dans le pire des cas, le nombre de comparaison effectuées est un $O(n^2)$.
3. Qu'en est-il dans le meilleur des cas ?

Exercice 4.

1. Proposer une variante sur les tableaux.
2. Implémenter la variante en OCaml.