TD Info n°6

PCSI 2 Lycée Pasteur

18 octobre 2007

Le but de ce TD et du suivant est de s'intéresser aux algorithmes de tri. Le problème est fort simple : vous disposez d'un tableau rempli de n éléments (par exemples des réels, ou des noms) qqui ne sont pas classés dans l'ordre (croissant ou alphabétique, selon les cas). Il s'agit de déterminer un algorithme (et si possible d'en écrire une implémentation en Maple) qui fasse ce travail et essayer d'en déterminer la complexité, c'est-à-dire ici le nombre de comparaisons nécessaires pour trier le tableau. Nous allons pour cela étudier quelques algorithmes effectuant le tri d'un tableau et essayer de les comparer.

1 Tri par sélection

Une méthode très intuitive pour trier un tableau est la suivante : on commence par chercher le plus petit élément de notre tableau, et on le place en tête, puisqu'il sera le premier élément du tableau trié. On recommence avec les éléments restants. Déterminer le nombre de comparaisons nécessaires pour trier un tableau à n éléments par cette méthode. Écrire une procédure en Maple faisant ce travail.

2 Tri par insertion

Encore un algorithme intuitif et un peu naïf. On créée une copie du tableau et on y place le premier élément du tableau à trier. Puis on place le deuxième élément au bon endroit dans le nouveau tableau (devant le premier ou derrière selon qu'il est plus petit ou plus grand), on ajoute ensuite le troisième élément etc. Comme pour le tri par sélection, déterminer la complexité de cet algorithme et implémentez-le en Maple.

3 Tri à bulle

Le nom ridicule de cet algorithme tient à une comparaison assez douteuse avec un phénomène visible dans les aquariums : les plus grosses bulles remontent plus vite que les petites. Ici, c'est un peu la même chose : on fait « remonter » les plus gros éléments en premier. En pratique, on effectue l'algorithme suivant : on parcourt le tableau à trier de gauche à droite, en comparant chaque élément à l'élément qui le suit. Si l'élément en question est plus grand que son successeur, on les échange. Une fois le parcours terminé, on recommence, mais en se dispensant de comparer les deux derniers éléments de tableau (pourquoi ?). Puis on continue à faire des parcours en diminuant d'un le nombre de comparaisons à chaque fois. Vérifier que cet algorithme fonctionne, calculer sa complexité et en écrire une implémentation en Maple.

La semaine prochaine, on s'intéressera à des algorithmes plus complexes, mais nettement plus efficaces.