

TD n°1

Dans les exercices suivants, on s'intéresse à des ensembles de n entiers tous distincts rangés dans un tableau $T[1], \dots, T[n]$. Les algorithmes considérés ici effectuent des **affectations** et leur seul critère de décision est la **comparaison** de deux éléments ($=$ et $<$). En aucun cas ils ne peuvent effectuer des opérations arithmétiques, comme l'addition ou la multiplication.

1 Maximum de n entiers

Question 1.1 Écrire un algorithme (naïf!) qui calcule le maximum de n entiers. Quelle en est la complexité (en nombre de comparaisons effectuées, en nombre d'affectations effectuées, dans le pire des cas, le meilleur) ?

Question 1.2 Quelle est la complexité en moyenne en nombre d'affectations effectuées, en nombre de comparaisons effectuées ?

Indications pour le calcul des affectations en moyenne : soit $P_{n,k}$ le nombre de permutations σ de $\{1, \dots, n\}$ telles que sur la donnée $T[1] = \sigma(1), \dots, T[n] = \sigma(n)$ l'algorithme effectue k affectations. Donner une relation de récurrence pour $P_{n,k}$. Soit $G_n(z) = \sum P_{n,k} z^k$. Montrer que $G_n(z) = z(z+1) \cdots (z+n-1)$. En déduire le résultat.

Question 1.3 L'algorithme que vous avez proposé est-il optimal pour la complexité en comparaisons dans le pire des cas ?

2 Plus petit et plus grand

Dans l'exercice suivant, on ne s'intéresse plus qu'à la **complexité dans le pire des cas et en nombre de comparaisons** des algorithmes.

Question 2.1 On s'intéresse maintenant au calcul (simultané) du maximum et du minimum de n entiers. Donner un algorithme naïf et sa complexité.

Question 2.2 Une idée pour améliorer l'algorithme est de regrouper *par paires* les éléments à comparer, de manière à diminuer ensuite le nombre de comparaisons à effectuer. Décrire un algorithme fonctionnant selon ce principe et analyser sa complexité.

Question 2.3 Montrons l'optimalité d'un tel algorithme en fournissant une borne inférieure sur le nombre de comparaisons à effectuer. Nous utiliserons la méthode de l'*adversaire*.

Soit A un algorithme qui trouve le maximum et le minimum. Pour une donnée fixée, au cours du déroulement de l'algorithme, on appelle *novice* (N) un élément qui n'a jamais subi de comparaisons, *gagnant* (G) un élément qui a été comparé au moins une fois et a toujours été supérieur aux éléments auxquels il a été comparé, *perdant* (P) un élément qui a été comparé au moins une fois et a toujours été inférieur aux éléments auxquels il a été comparé, et *moyens* (M) les autres. Le nombre de ces éléments est représenté par un quadruplet d'entiers (i, j, k, l) qui vérifient bien sûr $i + j + k + l = n$.

Donner la valeur de ce quadruplet au début et à la fin de l'algorithme. Exhiber une stratégie pour l'adversaire, de sorte à maximiser la durée de l'exécution de l'algorithme. En déduire une borne inférieure sur le nombre de tests à effectuer.