Option Informatique en Spé MP et MP*
Calcul simultané du min et du max

▶ Soit $E = \{x_1, \ldots, x_n\}$ un ensemble ordonné ; nous voulons en déterminer le plus petit élément m et le plus grand M, en effectuant le moins possible de comparaisons.

▶ Considérons l'algorithme suivant :

- préparer $p = \lfloor n/2 \rfloor$ paires d'éléments de E ; si n est impair, il reste un élément ξ ;
- comparer les deux membres de chaque paire ; ceci nous coûte p comparaisons et nous fournit une liste $(m_i)_{1 \leqslant i \leqslant p}$ de «petits» et une liste $(M_i)_{1 \leqslant i \leqslant p}$ de «grands» ;
- m est le plus petit des m_i ; on peut le déterminer au prix de $p-1$ comparaisons ;
- on peut de même déterminer M au prix de $p-1$ comparaisons ;
- si n est impair, comparer l'élément restant à m et (au besoin) à M.

▶ Le coût de cet algorithme est donné par les formules $C(2p) = 3p-2$ et $C(2p+1) = 3p$; un peu de réflexion nous donne $\boxed{C(n) = \lceil 3n/2 \rceil - 2}$ Nous allons montrer qu'il est impossible de faire mieux.

▶ Après une comparaison entre éléments de E, nous dirons que le plus petit a *perdu* et que le plus grand a *gagné*. Observons que E est la réunion disjointe de quatre parties, qui évoluent au cours du déroulement de l'algorithme :

- l'ensemble A des éléments qui n'ont encore jamais gagné ni perdu (car ils n'ont encore participé à aucune comparaison) ;
- l'ensemble B des éléments qui ont perdu au moins une fois, mais n'ont jamais gagné ;
- l'ensemble C des éléments qui ont gagné au moins une fois, mais n'ont jamais perdu ;
- l'ensemble D des éléments qui ont perdu au moins une fois et gagné au moins une fois.

Notons a, b, c et d les cardinaux respectifs de A, B, C et D. Au départ, $a = n$ et $b = c = d = 0$. Lorsque l'algorithme se termine, $a = 0$, $b = c = 1$ et $d = n-2$: $B = \{m\}$ et $C = \{M\}$.

▶ Appelons *saut* le passage d'un élément de E de l'une des parties à une autre ; il existe en fait quatre types de sauts : $A \to B$, $A \to C$, $B \to D$ et $C \to D$. L'algorithme devra donc effectuer au moins $2n-2$ sauts : n pour faire sortir tous les éléments de A, et $n-2$ pour amener dans D tous les éléments sauf le min et le max.

▶ Observations :

- une comparaison entre deux éléments de A permet d'effectuer deux sauts ($A \to B$ et $A \to C$) ; nous dirons qu'elle est *forte*.
- une comparaison entre deux éléments de B (ou entre deux éléments de C) permet d'effectuer un saut ($B \to D$ ou $C \to D$) ; une comparaison entre un élément de A et un élément de $B \cup C$ permet d'effectuer *au moins* un saut ($A \to B$ ou $A \to C$) ; nous qualifierons ces comparaisons de *faibles* ;
- le résultat de toute autre comparaison est imprévisible.

▶ Terminons la discussion : l'algorithme peut effectuer au plus p comparaisons fortes, ce qui donne déjà $2p$ sauts ; il lui restera à effectuer $2n-2p-2$ sauts au prix de $2n-2p-2$ comparaisons faibles. Le coût total s'élève donc au moins à :

$$2n-p-2 = n+n-\lfloor n/2 \rfloor - 2 = n + \lceil n/2 \rceil - 2 = \lceil 3n/2 \rceil - 2$$

Commentaire

C'est l'exercice 16 du chapitre 5.3.3 du *Art of Computer Programming*. Donald KNUTH en attribue l'idée à Ira Sheldon POHL.

FIN