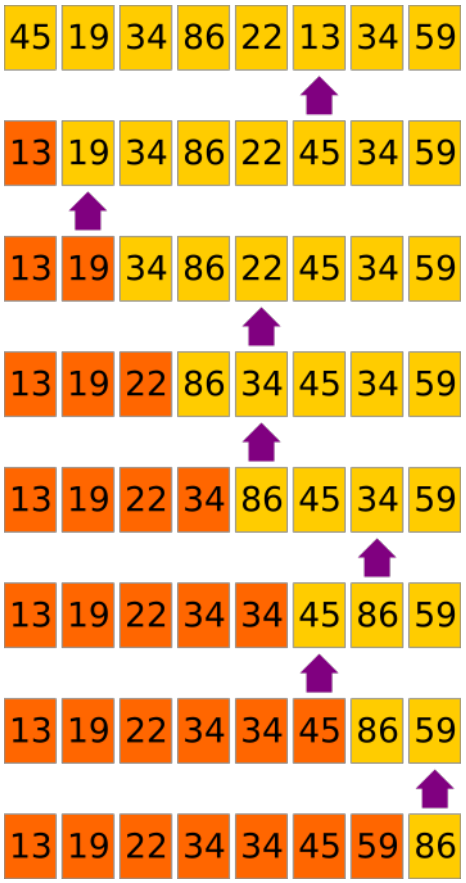
Tri par sélection – Correction

Énoncé



Tout le long de l'algorithme, on considère deux zones du tableau. L'une à gauche – sur fond orange dans l'exemple – et l'autre à droite – sur fond jaune dans l'exemple.

L'algorithme est constitué principalement d'une boucle répétitive. À chaque exécution de cette boucle, on sélectionne le plus petit élément parmi les jaunes et on le déplace à la suite de la zone orange. La zone orange s'en trouve augmentée d'un élément –ce qui diminue la jaune.

La boucle s'arrête quand la zone jaune ne compte plus qu'un seul élément. Le tableau est alors trié.

Questions : Justifiez que

a) la boucle principale arrive forcément à sa fin.

b) les items de la zones oranges sont toujours inférieurs ou égaux à ceux de la zone jaune,

c) les items de la zone orange sont dans l'ordre

d) le tableau est dans l'ordre à la fin de l'exécution

Correction

a) À chaque répétition de la boucle principale, on prend un élément de la zone jaune pour le mettre dans la zone orange. Donc la zone jaune perd un élément à chaque répétition.

Au début de l'algorithme, la zone jaune contient un nombre fini d'éléments.

Puisque la zone jaune part d'un nombre fini et qu'on le diminue de 1 à chaque répétition, il faudra un nombre fini de répétitions pour que la zone jaune n'ait plus qu'un élément ou moins.

On a dit que la boucle principale se répète tant que la zone jaune a plus d'un élément. On voit donc qu'il finira toujours par arriver le moment où ce ne sera plus le cas et donc que la boucle s'arrêtera.

b) Au début, la zone orange est vide. On peut donc affirmer qu'au début tous les éléments de la zone orange sont tous inférieurs aux éléments de la zone jaune.

Quand on sélectionne un élément dans la zone jaune, il est le plus petit. Il est donc inférieur à tous les autres éléments de la zone jaune.

Quand l'élément sélectionné passe dans la zone orange, cet élément est donc bien plus petit que tous les éléments restants dans la zone jaune.

Puisque au début de l'algorithme, la zone orange est vide et que tous les éléments qu'on y ajoute sont inférieurs à tous les éléments de la zone jaune, alors on peut affirmer que tous les éléments de la zone orange sont inférieurs à tous les éléments de la zone jaune.

c) Au début, la zone orange est vide, donc elle est triée.

Quand on ajoute un élément dans la zone orange, cet élément vient de la zone jaune. Or on vient de dire qu'à chaque instant, les éléments de la zone jaune sont plus grands que ceux de

la zone orange. Donc l'élément ajouté dans la zone orange est plus grand que tous ceux déjà présents dans la zone orange.

Puisque l'élément ajouté est le plus grand de la zone orange et qu'on le place en dernier, il préserver l'ordre.

La zone orange est donc toujours triée.

d) À la fin de l'algorithme, on a d'une part une zone orange qui est triée (puisque la zone orange l'est toujours). Il reste aussi un élément jaune qui est seul, plus grand que ceux de la zone orange et se trouve en dernier. Donc le tableau est bien trié à la fin.

Complexité

Prenons l'exemple qui contient $n = 8$ éléments.

- Au premier tour de boucle, on doit sélectionner le plus petit parmi 8, on imagine qu'il faut parcourir les 8 valeurs, donc on va faire 8 fois la même chose.
- Au bout de ce temps, il faut permuter la valeur trouvée.
- Ensuite on doit chercher le plus petit parmi 7, et faire une inversion.
- Chercher le plus petit parmi 6 puis inversion.
- ...

Si je note T_{petit} le temps élémentaire pour examiner chaque élément quand je cherche le plus petit et T_inv le temps d'une inversion, j'obtiens quelque chose comme :

$$T_{total} = 8T_{petit} + T_{inv} + 7T_{petit} + T_{inv} + \cdots + 2T_{petit} + T_{inv}$$

Soit encore

$$T_{total} = (8 + 7 + 6 + 5 + 4 + 3 + 2)T_{petit} + 7T_{inv}$$

Si au lieu d'avoir 8, on avait en général n, cela donnerait :

$$T_{total} = (n + \cdots + 2)T_{petit} + (n - 1)T_{inv}$$

Les maths me permettent de savoir :

$$T_{total} = \frac{n(n+1) - 2}{2}T_{petit} + (n - 1)T_{inv}$$

Quand on s'intéresse à la complexité, on s'intéresse au cas n grand. $\frac{n(n+1)-2}{2} = \frac{1}{2}n^2 + \frac{1}{2}n - 1$. Quand n est grand, le terme en n^2 est prépondérant. Les autres sont négligeables. On peut donc dire que (pour n grand) :

$$T_{total} \approx \frac{1}{2}n^2 T_{petit} + nT_{inv}$$

Là encore, même si $T_{inv} > T_{petit}$, on sait que pour n grand, $\frac{1}{2}n^2 T_{petit}$ va devenir prépondérant et donc

$$T_{total} \approx \frac{1}{2}n^2 T_{petit}$$

Enfin, quand on étudie la complexité, ce qui nous intéresse n'est pas le temps de calcul par lui même, mais la façon dont le temps de calcul augmente quand la taille du problème augmente.

Par exemple ici, comment T_{total} augmente quand n est multiplié par 10 ? On voit que T_{total} sera alors multiplié par $100 = 10^2$. On dira donc que cet algorithme à une **complexité d'ordre** n^2.