Exercice 1 : Correction du tri rapide

Le but de cet exercice est de démontrer la correction de l'algorithme de tri rapide tel qu'il est rappelé ci-dessous.

Algorithme 1 : Partitionner(T, d, f, p)

Entrée : Un tableau T indicé par $[\![0, n]\!]$, trois entiers d, f et p tels que $0 \leqslant d \leqslant p \leqslant f < n$
Sortie : Un indice $q \in [\![d, f]\!]$ tel que $T[q]$ vaut finalement la valeur qu'avait $T[p]$ avant l'appel
Effet : Le sous-tableau $T[\![d, f]\!]$ est permuté de sorte que $\begin{cases} \forall k \in [\![d, q]\!],\ T[k] \leqslant T[q] \\ \forall k \in]\!]q, f[\![,\ T[k] > T[q] \end{cases}$

```
1  pivot ← T[p] ;
2  échanger T[p] et T[d] ;                          // on place le pivot au début
3  a ← d + 1 ;
4  b ← f ;
5  tant que a ⩽ b faire
6      si T[a] ⩽ pivot alors
7          a ← a + 1
8      sinon
9          échanger T[a] et T[b] ;
10         b ← b − 1 ;
11 échanger T[d] et T[a − 1] ;
12 retourner a − 1
```

Algorithme 2 : Procédure tri_rapide(T, d, f)

Entrée : Un tableau T indicé par $[\![0, n[\![$, deux entiers $d \leqslant 0$ et $f \leqslant n - 1$
Sortie : Aucune sortie, mais $T[\![d, f]\!]$ est triée par ordre croissant

```
1 si d < f alors
2     p ← choix_pivot(T, d, f)          // choix d'un entier entre d et f ;
3     q ← Partitionner(T, d, f, p) ;
4     tri_rapide(T, d, q − 1) ;
5     tri_rapide(T, q + 1, f)
```

Q. 1 Exécuter l'algorithme Partitionner sur le tableau T : `[0;1;5;3;4;2]` pour $d = 2$ et $f = 5$. Représenter l'état du tableau T à chaque étape ainsi que les valeurs des variables a et b.

Solution

...

Q. 2 Proposer un invariant♣ de boucle pour Partitionner qui précise les valeurs possibles et surtout le rôle des variables a et b.

Solution

$d + 1 \leqslant a \leqslant b \leqslant f\ \forall k \in [\![d + 1, a[\![,\ T[k] \leqslant \text{pivot}\ \forall k \in]\!]b, f]\!],\ T[k] > \text{pivot}$

Q. 3 Démontrer que l'invariant proposé est bien un invariant de boucle.

Solution

$d + 1 \leqslant a \leqslant b \leqslant f\ \forall k \in [\![d + 1, a[\![,\ T[k] \leqslant \text{pivot}\ \forall k \in]\!]b, f]\!],\ T[k] > \text{pivot}$

♣. celui-ci peut être composé de plusieurs propriétés

Q. 4 On admet que tout au long de l'algorithme le tableau T est une permutation du tableau initial, et que cette éventuelle transformation ne concerne que des indices compris dans l'intervalle $[\![d, f]\!]$ (donc la partie de T extérieure à cet intervalle n'est pas modifiée). À l'aide de l'invariant proposé à la question précédente démontrer que l'algorithme Partitionner est correct.

Solution

Q. 5 En admettant la correction de l'algorithme Partitionner, démontrer que l'algorithme Tri_rapide est correct.

Solution

Solution

0.1 Étude de correction du tri rapide

0.1.1 Correction de partitionner

Démontrons dans un premier temps que la fonction partitionner est correcte. Pour une entrée valide de l'algorithme (T, p, d, f), on appelle **valeur pivot** la valeur de $T[p]$ (avant l'appel donc).

Remarque 0.1

On remarque qu'à chaque étape on applique à T une transposition ou aucune transformation, donc tout au long de l'algorithme le tableau T est une permutation du tableau initial. De plus cette éventuelle transposition ne concerne que des indices compris dans l'intervalle $[\![d, f]\!]$ (ce fait est assuré par le premier point le l'invariant proposé ci-après), donc la partie de T extérieure à cet intervalle n'est pas modifiée.

Lemme 0.2

La fonction partitionner admet l'invariant de boucle suivant.

$$\mathcal{G} : \begin{cases} d \leqslant J \leqslant I \leqslant f \\ T[f] = \textit{la valeur du pivot} \\ \forall k \in [\![d, J-1]\!], T[k] \leqslant T[f] \\ \forall k \in [\![J, I-1]\!], T[k] > T[f] \end{cases}$$

Démonstration : • On commence l'exécution en plaçant la valeur pivot dans la case d'indice f, donc on a bien $T[f] =$ *la valeur du pivot*. De plus on a initialement$I = d$ et $J = d$, donc $d \leqslant J \leqslant I \leqslant f$ et les deux autres propriétés sont trivialement vraies (car quantifiées universellement sur l'ensemble vide).

• Notons $\underline{I}$, $\underline{J}$, $\underline{T}$ les valeurs de I, J et du contenu du tableau T au début d'une itération de boucle. Notons $\overline{I}$, $\overline{J}$, $\overline{T}$ les valeurs de I, J et du contenu du tableau T au début de l'itération de boucle suivante (ou à défaut à la fin de l'itération de la boucle considérée). Supposons que l'invariant $\mathcal{G}$ est vrai sur les valeurs avant l'itération de boucle. On a alors :

$$\begin{array}{l} d \leqslant \underline{J} \leqslant \underline{I} \leqslant f \\ \underline{T}[f] = \textit{la valeur du pivot} \\ \forall k \in [\![d, \underline{J}-1]\!], \underline{T}[k] \leqslant \underline{T}[f] \\ \forall k \in [\![\underline{J}, \underline{I}-1]\!], \underline{T}[k] > \underline{T}[f] \end{array}$$

On a $\underline{I} < f$, on a $\overline{I} = \underline{I} + 1$,

- Si $\underline{T}.(\underline{I}) \leqslant \underline{T}[f]$ alors :
  - $\forall k \notin \{\underline{I}, \underline{J}\}, \overline{T}[k] = \underline{T}[k]$
  - $\overline{T}[\underline{I}] = \underline{T}.(\underline{J})$
  - $\overline{T}[\underline{J}] = \underline{T}.(\underline{I})$
  - $\overline{J} = \underline{J} + 1$

  Soit alors $k \in [\![d, \overline{I} - 1]\!]$,
  - Si $k < \overline{J} - 1$ alors $d \leqslant k \leqslant \underline{J} - 1$ et donc, par invariant on a que $\overline{T}[k] = \underline{T}[k] \leqslant \underline{T}[f] = \overline{T}[f]$.
  - Si $k = \overline{J} - 1$ alors $k = \underline{J}$ et donc, par invariant : $\overline{T}[k] = \overline{T}[\underline{I}] \leqslant \underline{T}[f] = \overline{T}[f]$.
  - Sinon si $k < \overline{I} - 1$ ($\overline{J} \leqslant k < \overline{I} - 1$) et donc, par invariant on a que $\overline{T}[k] = \underline{T}[k] > \underline{T}[f] = \overline{T}[f]$.
  - Sinon ($k = \overline{I} - 1$) donc par invariant : $\overline{T}[k] = \overline{T}[\underline{I}] = \underline{T}.(\underline{J}) > \underline{T}[f] = \overline{T}[f]$

  De plus $d \leqslant \underline{J} \leqslant \overline{J}$, $\overline{J} = \underline{J} + 1 \leqslant \underline{I} + 1 = \overline{I}$, et $\overline{I} = \underline{I} + 1 \leqslant f$.
- Si $\underline{T}.(\underline{I}) > \underline{T}[f]$ alors :
  - $\forall k, \overline{T}[k] = \underline{T}[k]$
  - $\overline{J} = \underline{J}$

  Soit alors $k \in [\![d, \overline{I} - 1]\!]$,
  - Si $k \leqslant \overline{J} - 1$ alors $d \leqslant k \leqslant \underline{J} - 1$ et donc, par invariant on a $\overline{T}[k] = \underline{T}[k] \leqslant \underline{T}[f] = \overline{T}[f]$.
  - Sinon si $k < \overline{I} - 1$ ($\overline{J} \leqslant k < \overline{I} - 1$) et donc, par invariant on a $\overline{T}[k] = \underline{T}[k] > \underline{T}[f] = \overline{T}[f]$.
  - Sinon ($k = \overline{I} - 1$) donc par invariant : $\overline{T}[k] = \overline{T}[\underline{I}] > \underline{T}[f] = \overline{T}[f]$.

  De plus $d \leqslant \underline{J} = \overline{J}$, $\overline{J} = \underline{J} \leqslant \underline{I} + 1 = \overline{I}$, et $\overline{I} = \underline{I} + 1 \leqslant f$.

□

Proposition 0.3

Soit (T, d, f, p) une entrée valide de partitionner. Notons n la taille de T.
L'appel partitionner(T, d, f, p) produit un indice $q \in [\![d, f]\!]$, et en notant $\underline{T}$ (resp. $\overline{T}$) la valeur de T avant (resp. après) l'appel on a :
- *$\overline{T}$ est une permutation de $\underline{T}$;*
- *$\overline{T}[J] = \underline{T}[p]$;*
- *$\forall k \in [\![d, J-1]\!]$, $\overline{T}[k] \leqslant \overline{T}[J]$;*
- *$\forall k \in [\![J+1, f]\!]$, $\overline{T}[k] > \overline{T}[J]$;*
- *$\forall k \in [\![0, n-1]\!] \setminus [\![d, f]\!]$, $\overline{T}[k] = \underline{T}[k]$.*

Démonstration : Se démontre en s'appuyant sur le lemme précédent. □

0.1.2 Correction de tri_rapide

Théorème 0.4

Soit (T, d, f, p) une entrée valide de partitionner. Notons n la taille de T.
L'appel tri_rapide(T, d, f) est tel qu'en notant $\underline{T}$ (resp. $\overline{T}$) la valeur de T avant (resp. après) l'appel on a :
- *$\overline{T}$ est une permutation de $\underline{T}$;*
- *$\overline{T}_{[\![d,f]\!]}$ est trié ;*
- *$\forall k \in [\![0, n-1]\!] \setminus [\![d, f]\!]$, $\overline{T}[k] = \underline{T}[k]$.*

Remarque 0.5

Cela implique que $\overline{T}[\![d, f]\!]$ est une permutation de $\underline{T}[\![d, f]\!]$.

Démonstration : On prouve ce résultat par récurrence sur la valeur de $f - d$.

• Si $d = f$ alors $\overline{T} = \underline{T}$ et les propriétés sont trivialement vraies.

• Si $f - d > 0$, soit alors $p \in [\![f, d]\!]$. Notons T_1 la valeur de T après l'appel partitionner(T, d, f, p), on a alors i la valeur retournée par cet appel. Nous avons par la proposition précédente :

- T_1 est une permutation de T
- $T_1[i] = T[p]$
- $\forall k \in [\![d, i-1]\!], T_1[k] \leqslant T_1[i]$
- $\forall k \in [\![i+1, f]\!], T_1[k] > T_1[i]$
- $\forall k \in [\![0, n-1]\!] \setminus [\![d, f]\!], T_1[k] = T[k]$

Notons alors T_2 la valeur de T après l'appel récursif tri_rapide$(T, d, i-1)$, par induction nous avons que :

- T_2 est une permutation de T_1 ;
- $T_2[d..i-1]$ est trié ;
- $\forall k \in [\![0, n-1]\!] \setminus [\![d, i-1]\!], T_2[k] = T_1[k]$

Ainsi nous avons :

- T_2 est une permutation de T (par composée de permutations)
- $T_2[d..i]$ est trié (car $T_2[d..i-1]$ est une permutation de $T_1[d..i-1]$ qui sont tous inférieurs à $T_1[i]$)
- $\forall k \in [\![0, n-1]\!] \setminus [\![d, i-1]\!], T_2[k] = T_1[k]$

On conclut de même après l'appel tri_rapide$(t, i+1, f)$. □