

UE ALGO5 — TD2 — Séance 5 : Drapeau arc-en-ciel

Objectifs

À la fin de cette séance, vous devriez être capable de :

- adapter et généraliser des algorithmes vus en cours ;
- écrire des spécifications (pré- et post- conditions) de fonctions ;
- spécifier et prouver des invariants de boucle.

Exercice 1. Drapeau arc-en-ciel

On cherche dans ce TD à généraliser l’algorithme vu en cours du *Drapeau Hollandais*, dans le cas où il y a P couleurs distinctes (au lieu de 3).

L’ordre souhaité pour les couleurs est donné par un tableau C .

Q1. Compléter la spécification de la procédure DrapeauAEC :

```

1 Couleur : type abstrait
3 DrapeauAEC(C,P,T,N)
  { Données :
5   P, N : entiers
   C : tableau sur [0..P-1] de couleurs
7   T : tableau sur [0..N-1] de couleurs
   Pré-condition : ...
9   Post-condition : ...
  }
```

Corrigé —

Pré-condition: il faut expliciter et formaliser le fait que :

- toutes les couleurs de C sont différentes (1)
- toutes les valeurs de T sont dans C (2)

$$\forall i \in [0, P - 1], \forall j \in [0, P - 1], i \neq j \Rightarrow C[i] \neq C[j] \tag{1}$$

$$\forall i \in [0, N - 1], \exists j \in [0, P - 1], T[i] = C[j] \tag{2}$$

Post-condition: il est temps de réfléchir à ce que va construire l’algorithme. Pour le drapeau hollandais, la post-condition peut s’exprimer à l’aide de 2 indices (pour délimiter 3 tranches dans le tableau). Ici il faut utiliser $P - 1$ indices.

Il existe $i(1), \dots, i(P - 1)$ tels que :

$$\begin{array}{|c|c|c|c|} \hline i(0) & i(1) & i(2) & i(P-1) \\ \hline = C[0] & = C[1] & & = C[P-1] \\ \hline \end{array}$$

D’où :

- $\exists i(0), \dots, i(P)$ t.q.
- $\forall j \in [0, P - 1], i(j) \leq i(j + 1)$
- $i(0) = 0$
- $i(P) = N$

— $\forall j \in [0, N - 1], \exists k \in [1, P], i(k - 1) \leq j < i(k)$ et $T[j] = C[k]$
 ... et bien sûr, le tableau T est une permutation du tableau initial (vu en TD...).

Q2. Écrire le schéma et l'invariant de la boucle principale de l'algorithme.

Corrigé —

On utilise donc un tableau $I[0..P]$ pour stocker les indices délimitant les tranches du tableau.

On peut se baser sur la post-condition pour construire l'invariant de la boucle principale :

$I[0]$	$I[1]$	$I[2]$	$I[P - 1]$	$I[P]$	$N - 1$
$= C[0]$	$= C[1]$		$= C[P - 1]$	non traités	

Le corps de l'itération consiste donc à insérer l'élément à l'indice $I[P]$ à la bonne place. Comme pour le drapeau hollandais, il n'est pas nécessaire de décaler tous les éléments : on peut faire des permutations successives de l'élément à placer avec les éléments situés aux indices $I[j]$.

On peut donc exprimer l'invariant de l'itération interne (celle qui va placer l'élément x) :

$I[0]$	$I[1]$	$I[2]$	$I[j - 1]$	$I[j]$	$I[P - 1]$	$I[P]$	$N - 1$
$= C[0]$	$= C[1]$		$= C[j]$	x	$= C[P - 1]$	non traités	

Avec $\exists k \in [0, P - 1]$ t.q. $x = C[k]$ et $k \leq j$.

Q3. Écrire l'algorithme complet.

Corrigé —

Une solution possible, obtenue à partir de l'invariant précédent, et fortement inspirée du drapeau hollandais :

```

DrapeauAEC(C, P, T, N)
  I : tableau sur [0..P] d'entiers
  pour j de 0 à P faire
    | I[j] := 0
  fin
  tant que I[P] ≠ N faire
    // Décalages successifs
    j := P
    tant que T[I[j]] ≠ C[j] faire
      // Échange de x avec le premier élément de la tranche courante
      tmp := T[I[j]]
      T[I[j]] := T[I[j - 1]]
      T[I[j - 1]] := tmp

      I[j] := I[j] + 1
      j := j - 1
    fin
    I[j] := I[j] + 1
  fin

```

D'autres solutions existent : avec un tableau auxiliaire ; en plusieurs passes ; en se basant sur un comptage des couleurs...

Q4. Évaluer la qualité de votre solution :

- quel est sa complexité en temps au pire?
 - quel est son coût en mémoire (en plus de l'espace occupé par les tableaux T et C)?
 - combien de fois demande-t-elle d'évaluer la couleur d'un élément de T ?
-

Corrigé —

- Quelle que soit l'idée directrice retenue, on doit pouvoir arriver à une solution en temps $\mathcal{O}(n \times p)$.
 - Le coût en mémoire dépend principalement de l'utilisation ou non d'un tableau auxiliaire pour recopier les éléments de T . Il est ici plus intéressant (et possible!) de travailler en place, mais on aura toujours besoin d'un tableau de P indices ou de quelque chose d'équivalent.
 - Le nombre de calculs de couleurs dépend bien sûr de la solution retenue; il est clair que cette quantité ne peut pas descendre en-dessous de N , borne que l'algorithme proposé ci-dessus permet d'atteindre.
-