Drapeau hollandais

Concepts : Analyse de coût
Méthodes : Décomposition du coût, ordres de grandeur

Présentation

On dispose d'un tableau de n éléments, chaque élément est coloré avec une des 3 couleurs bleu, blanc ou rouge. L'objectif est de réorganiser le tableau de manière à ce que les éléments bleus soient sur la partie gauche du tableau les éléments blancs au centre et les rouges en fin de tableau (voir la figure 1).

FIGURE 1 – Exemple de tableau d'entrée et de tableau résultat

L'objectif est de réaliser ce réarrangement en utilisant un minimum de mémoire supplémentaire, c'est à dire que l'on peut avoir un peu de mémoire pour réaliser l'échange et gérer quelques indices.

Algorithme

DRAPEAU(T)
Données : Un tableau T de N éléments colorés (couleur C_1, C_2 et C_3)
Résultat : Le tableau T contient les mêmes éléments mais rangés par couleur croissante

$i_1 = 1$ $i_2 = N$ $i_3 = N$
// i_k indice de la place du prochain élément de couleur C_k
while $i_1 \leqslant i_2$
1 **Assertion :** T contient une permutation des éléments du tableau initial et les éléments de 1 à $i_1 - 1$, de $i_2 + 1$ à i_3 et de $i_3 + 1$ à N sont de couleurs respectives C_1, C_2 et C_3
switch Couleur $(T[i_1])$ **do**
case C_1
$i_1 = i_1 + 1$ // l'élément est en place
case C_2
Échange (i_1, i_2) // on le place en bonne position
$i_2 = i_2 - 1$
case C_3
Échange (i_1, i_2) // on fait une permutation circulaire
Échange (i_2, i_3) // pour libérer la case entre C_2 et C_3
$i_2 = i_2 - 1$; $i_3 = i_3 - 1$

Algorithme 1: Algorithme du Drapeau Hollandais

Drapeau hollandais

Preuve

La preuve se fait en 2 étapes. La première consiste à montrer que l'assertion ligne 1 est vraie à chaque passage dans l'itération et également à la fin de l'itération. Le deuxième étape consiste à montrer que l'algorithme se termine, c'est à dire que la condition $(i_1 \leqslant i_2)$ est fausse au bout d'un temps fini.

Étape 1 : La première partie de l'assertion, "le tableau T contient une permutation des éléments du tableau initial" est garantie par le fait que les seules opérations de modification du tableau sont faites par la procédure **Échange** qui laisse invariant l'ensemble des éléments contenus dans le tableau. La deuxième partie de l'assertion est également propagée. En effet elle est vraie au début de la première itération (aucun élément n'est placé et il n'y a pas d'éléments de 1 à $i_1 - 1$, ni de $i_2 + 1$ à i_3 et ni de $i_3 + 1$ à N). Supposons que cette assertion soit vraie en début d'itération, alors en étudiant chacun des 3 cas séparément il est clair que l'assertion est vraie en fin d'itération. Une illustration est proposée par la figure 2. À la fin de la dernière itération, on a $i_2 = i_1 - 1$, il n'y a

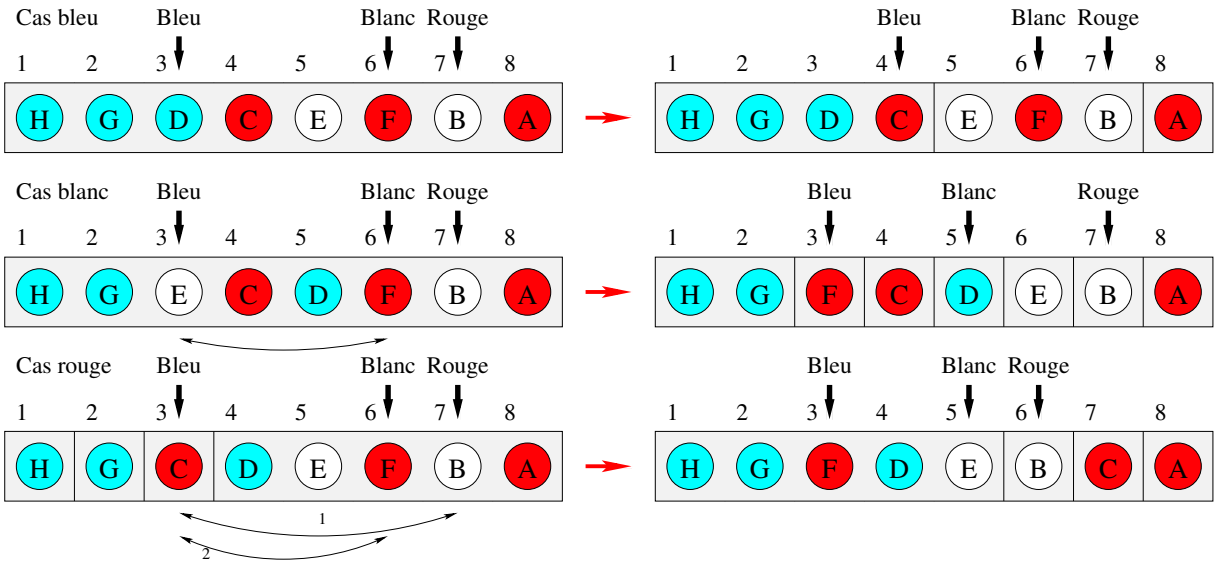

FIGURE 2 – Les 3 cas de couleur

donc plus d'éléments à consulter et les éléments sont rangés par couleur croissante.

Étape 2 : L'algorithme se termine car à chaque étape l'écart entre la valeur de i_1 et la valeur de i_2 est diminuée de 1. Par conséquent on aura au plus N itérations.

Complexité

Pour évaluer le coût de cet algorithme en nombre d'appels à des opérations "primitives" on décompose les itérations, le coût total sera la somme des coûts de chaque itérations.

Le coût en nombre d'appels à la fonction `Couleur` correspond au nombre d'itérations, c'est à dire N

$$\text{Coût}_{\texttt{Couleur}}(N) = N.$$

On évalue la couleur de chaque élément du tableau une et une seule fois.

Le coût en nombre d'appels à la fonction `Échange` correspond à la somme du nombre d'échanges réalisés à chaque itération. Or le nombre d'échanges à chaque itérations est variable (dépend des données en entrée), il peut valoir 0, 1 ou 2 selon les cas. On en déduit que

$$0 \leqslant \text{Coût}_{\texttt{Échange}}(N) \leqslant 2N.$$

Drapeau hollandais

Le meilleur cas est obtenu lorsque chaque itération passe par le cas 1 du de l'instruction switch, c'est à dire que le tableau est rempli d'éléments bleus. Le pire cas est obtenu lorsque chaque itération passe par le cas 3 du de l'instruction switch, c'est à dire que le tableau est rempli d'éléments rouges.

La complexité de l'algorithme en nombre d'échanges est donc au pire en $\mathcal{O}(N)$ et au mieux en $\mathcal{O}(1)$

Exercices

1. Écrire un algorithme qui résout le même problème mais avec 2 couleurs seulement.
2. Peut-on économiser quelques échanges ? Cela change-t-il l'ordre de grandeur de la complexité au pire ?
3. Implémenter l'algorithme et évaluer expérimentalement le coût moyen de l'algorithme lorsque les couleurs sont attribuées uniformément (on attribue indépendamment à chaque élément une couleur uniformément choisie dans $\{1, 2, 3\}$).
4. Écrire un programme de tri récursif qui s'appuie sur une décomposition en 3.

Historique

Ce problème aurait été proposé par E.W Dijkstra [1] chap. 14 qui le détiendrait de W.H.J. Feijen. E.W. Dijkstra est l'un des fondateurs de la science informatique et à l'origine d'un grand nombre d'algorithmes (ex. plus court chemin dans un graphe). Une biographie de Dijkstra est accessible à l'url `http://www.gap-system.org/~history/Biographies/Dijkstra.html`.

Références

[1] Edsger W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[2] Robert Sedgewick and Kevin Wayne. *Algorithms (4th Edition)*. Addison-Wesley Professional, 2011.