Complexité
TD 3 - Correction
Master 1 Informatique

Serge Miguet, Guillaume Metzler, Tess Masclef
Institut de Communication (ICOM)
Université de Lyon, Université Lumière Lyon 2

serge.miguet@univ-lyon2.fr **guillaume.metzler@univ-lyon2.fr**

tess.masclef@univ-lyon2.fr

Le drapeau Hollandais

On considère un tableau de n éléments, chaque élément étant colorié avec l'une des trois couleurs bleu, blanc ou rouge. Le but du problème est de réorganiser le tableau en ne procédant qu'à des échanges de deux éléments, de manière à ce que les éléments bleus soient en début de tableau, les éléments blancs au centre et les rouges en fin de tableau.

Algorithmes permettant de résoudre ce problème La solution d'un tel problème est à la base d'algorithmes de tri tel que "quicksort". On souhaite regrouper les éléments bleus, blancs et rouges.

Cas bicolore On dispose d'un tableau de n éléments, chaque élément est coloré avec 2 couleurs, bleu ou rouge. l'objectif est de réorganiser le tableau de manière à ce que les éléments bleus soient sur la partie gauche et les rouges sur la partie de droite.

On utilise un tableau de taille n dont les valeurs appartiennes à $\{Bleu, Rouge\}$. Soient i_r et i_b deux indices indiquant la plage des "cases indéterminées". Si i_b correspond à un case bleu, i_b est incrémenté de 1(la plage des cases bleues s'étend), sinon, on échange les contenus du tableau aux emplacements i_b et i_r et i_r est décrémenté de 1 (la plage des cases rouges s'étend). L'algorithme 1 décrit cette procédure.

Complexité : elle s'obtient en nombre d'échange. Si toutes les cases sont à leur place (donc si toutes les bleu sont en début de tableau), dans le meilleur des cas, le nombre d'échange est égal à 0, la complexité est donc de $\mathcal{O}(1)$. Dans le pire des cas, si aucune case n'est à sa place, le nombre d'échange est égal au nombre d'itération, soit n, la complexité est donc de $\mathcal{O}(n)$.

Algorithm 1: Algorithme : cas bicolore

Inputs : Un tableau T

Outputs: Un tableau T ordonné

$i_b = 0$
$i_r = n - 1$
while $i_b < i_r$ **do**
    if $T[i_1] =$ "bleu" **then**
        $i_b = i_b + 1$
    else
        Permuter $T[i_b]$ et $T[i_r]$
        $i_r = i_r - 1$
    end if
end while=0

Cas tricolore On dispose d'un tableau de n éléments que l'on nomera T, chaque élément est coloré avec 3 couleurs, bleu (représenté par un 0), blanc (représenté par un 1) ou rouge (représenté par un 2). L'objectif est de réorganiser le tableau de manière à ce que les éléments bleus soient sur la partie gauche, les éléments blancs au centre et les rouges sur la partie de droite.

On a donc :

0	1	1	2	0	1	2	1	2	0	0

Et on veut :

0	0	0	0	1	1	1	1	2	2	2

On va considérer un élément qui sera présent dans le milieu du tableau, ici un 1, que l'on va sélectionner comme un 'pole fixe'. C'est-à-dire que peu importe le 0 que l'on prendra dans le tableau on le mettra à gauche et peu importe le 2 que l'on prendra on le mettra à droite, ainsi automatiquement les 1 se retrouveront dans le centre du tableau.

Soient i_{min}, i_{max} et i_{mid} des indices indiquant la plage de "case indéterminées". On initialise i_{min} à 0, i_{max} à $n-1$ et i_{mid} à 0.

Prenons l'exemple du tableau ci-dessus où $n = 11$:

i_{min}	i_{max}	i_{mid}	T[i_{mid}]	
0	10	0	0	Le premier élément du tableau est 0, celui étant déjà à l'extrème gauche (c'est comme-ci on le switchait avec lui-même), on augment alors de 1 les indices i_{min} et i_{mid}
1	10	1	1	Le deuxième élément étant un 1 considéré comme un 'pole fixe', on n'y touche pas et on augmente l'indice i_{mid} de 1
1	10	2	1	Le troisième élément étant un 1, on augmente l'indice i_{mid} de 1
1	10	3	2	Le quatrième élément est un 2, on souhaite donc l'envoyer à droite du tableau, on va donc switcher le T[i_{mid}] élément avec le T[i_{max}] élément, ainsi on retire 1 à l'indice i_{max}

A ce niveau notre tableau ressemble à :

0	1	1	2 ↑	0	1	2	1	2	0	0
0	1	1	0 ↑	0	1	2	1	2	0	2

On a échangé $T[3]$ avec $T[10]$

On a ensuite :

i_{min}	i_{max}	i_{mid}	T[i_{mid}]	
1	9	3	2	Le quatrième élément du tableau est 0, on va donc switcher le T[i_{mid}] avec le T[i_{min}] élément, on ajoute alors 1 à i_{mid} et i_{min}

0	1	1	0 ↑	0	1	2	1	2	0	2
0	0	1	1	0 ↑	1	2	1	2	0	2

On a échangé $T[3]$ avec $T[1]$

On a ensuite :

i_{min}	i_{max}	i_{mid}	T[i_{mid}]	
2	9	4	0	Le cinquième élément du tableau est 0, on va donc switcher le T[i_{mid}] avec le T[i_{min}] élément, on ajoute alors 1 à i_{mid} et i_{min}

0	0	1	1	0 ↑	1	2	1	2	0	2
0	0	0	1	1	1 ↑	2	1	2	0	2

On a échangé $T[4]$ avec $T[2]$

Après

i_{min}	i_{max}	i_{mid}	T[i_{mid}]	
3	9	5	1	Le sixième élément du tableau est 1, on y touche pas et ajoute 1 à i_{mid}
3	9	6	2	Le septième élément du tableau est 2, on souhaite donc l'envoyer à droite du tableau, on va donc switcher le T[i_{mid}] élément avec le T[i_{max}] élément, ainsi on retire 1 à l'indice i_{max}

0	0	0	1	1	1	2 ↑	1	2	0	2
0	0	0	1	1	1	0 ↑	1	2	2	2

On a échangé $T[6]$ avec $T[9]$

Et on continue :

i_{min}	i_{max}	i_{mid}	T[i_{mid}]	
3	8	6	2	Le septième élément du tableau est 0, on va donc switcher le T[i_{mid}] avec le T[i_{min}] élément, on ajoute alors 1 à i_{mid} et i_{min}

0	0	0	1	1	1	0 ↑	1	2	2	2
0	0	0	0	1	1	1	1 ↑	2	2	2

On a échangé $T[6]$ avec $T[3]$

On a que $i_{min} = 4$, $i_{mid} = 7$ et $i_{max} = 8$ et le tableau est parfaitement rangé comme souhaité.

L'algorithme 2 décrit cette procédure.

Algorithm 2: Algorithme : cas bicolore

Inputs : Un tableau T

Outputs: Un tableau T ordonné

$i_b = 0$
$i_w = 0$
$i_r = n - 1$
while $i_b < i_r$ **do**
  if $T[i_w] =$ "blanc" **then**
    $i_w = i_w + 1$
  else if $T[i_w] =$ "bleu" **then**
    Permuter $T[i_b]$ et $T[i_w]$
    $i_w = i_w + 1$
    $i_b = i_b + 1$
  else
    Permuter $T[i_w]$ et $T[i_r]$
    $i_r = i_r - 1$
  end if
end while$=0$