Rangements

Objectifs

- Comprendre et analyser des algorithmes de rangement par partitions
- Savoir écrire l'algorithme du drapeau hollandais
- Savoir écrire l'algorithme du rangement binaire

Introduction

L'objectif est de ranger une collection classée en 3 catégories sans se servir de tableau annexe et en conservant l'ordre des éléments.

Ce problème a été posé en 1970 par **Edsger Dijkstra**, informaticien néerlandais. Il peut être résolu naïvement en triant le tableau classiquement.

Mais il propose un autre algorithme en utilisant trois indices (un pour chaque catégorie) plus performant.

C'est celui que nous allons étudier dans ce chapitre.

Algorithme du drapeau hollandais

Principe

On veut trier un tableau comportant 3 couleurs uniquement : **Bleu (B)**, **Blanc (W)** et **Rouge (R)**, de telle sorte que le tableau final soit Bleu-Blanc-Rouge comme le drapeau hollandais.

On va utiliser 3 indices :

- b : indice de la fin du bleu +1
- w : indice de la fin du blanc +1
- r : indice du début du rouge -1

On va parcourir le tableau avec l'indice w et suivant la couleur, on échangera les valeurs pour qu'elles soient dans le bon groupe et on mettra à jour les indices correspondants.

Structure du tableau pendant l'algorithme

Le tableau est divisé en 4 zones :

```
[Bleu | Blanc | ? (non traité) | Rouge]
  ^       ^       ^                ^
  1       b       w                r
```

- De 1 à $b-1$: éléments **Bleus** (triés)
- De b à $w-1$: éléments **Blancs** (triés)
- De w à r : éléments **non encore traités**
- De $r+1$ à n : éléments **Rouges** (triés)

Principe lors du parcours :

- Si `T[w] = Blanc` : on incrémente simplement w (le blanc est déjà bien placé)
- Si `T[w] = Rouge` : on échange `T[w]` avec `T[r]`, puis on décrémente r
- Si `T[w] = Bleu` : on échange `T[w]` avec `T[b]`, puis on incrémente b et w

Exemple de traitement

Soit le tableau initial : `[R, B, W, R, B, W, R, B]`
État initial : - $b = 1$, $w = 1$, $r = 8$ - `T[1] = R` → Échanger avec `T[8]` → `[B, B, W, R, B, W, R, R]`, $r = 7$
Après quelques itérations : - Les bleus migrent vers la gauche - Les blancs restent au milieu - Les rouges migrent vers la droite
État final : - `[B, B, B, W, W, R, R, R]`

Pseudo-code

```
ALGORITHME DrapeauHollandais(T):
DONNEES
  T : tableau de couleur de taille n
VARIABLES:
  b, w, r : entiers
DEBUT
  b ← 1
  w ← 1
  r ← n
  TQ w   r FAIRE
    SI T[w] = Blanc ALORS
```

```
      w ← w+1
    SINON
      SI T[w] = Rouge ALORS
        Echanger(T, w, r)
        r ← r-1
      SINON
        Echanger(T, w, b)
        b ← b+1
        w ← w+1
      FSI
    FSI
  FTQ
FIN
```

Arrêt

La condition d'arrêt de la boucle est équivalente à $r - w < 0$. Or $r - w$ va prendre les valeurs d'une suite strictement décroissante car à chaque passage de boucle, soit w augmente, soit r diminue, donc dans tous les cas $r - w$ diminue. Ainsi $r - w$ sera inférieur à 0 à un certain moment.

Validité

À l'itération k, notons $\mathcal{P}(k)$: "le sous-tableau de T de 1 à $b-1$ est Bleu, de b à $w-1$ est Blanc et de $r+1$ à n est Rouge."

- **Initialisation :** Au début de l'algorithme, les trois sous-tableaux sont vides (car $b = 1$, $w = 1$, $r = n$), donc $\mathcal{P}(0)$ est vraie.
- **Hérédité :** Supposons que $\mathcal{P}(k)$ soit vraie pour k fixé. Le passage dans la boucle nous offre trois possibilités :
  - **Cas 1 :** $T[w]$ **est Blanc.** Dans ce cas, comme le sous-tableau de b à $w-1$ était blanc, celui de b à w l'est aussi. L'instruction `w ← w+1` permet de réécrire que le sous-tableau de b à $w-1$ est blanc.
  - **Cas 2 :** $T[w]$ **est Rouge.** Dans ce cas, on échange `T[w]` et `T[r]`, ce qui veut dire que `T[r]` est Rouge donc comme le sous-tableau de T de $r+1$ à n était rouge, alors celui de r à n aussi. L'instruction `r ← r-1` permet de réécrire que le sous-tableau de $r+1$ à n est Rouge.
  - **Cas 3 :** $T[w]$ **est Bleu.** Dans ce cas, on échange `T[w]` avec `T[b]` qui devient bleu et donc le sous-tableau de 1 à b est bleu (car il était déjà bleu de 1 à $b-1$) et l'instruction `b ← b+1` permet de réécrire que le sous-tableau de 1 à $b-1$ est bleu. De la même façon, `T[w]` devient blanc (car le tableau de b à $w-1$ était blanc) et

donc le sous-tableau de b à w est blanc et l'instruction `w ← w+1` permet de réécrire que le sous-tableau de b à $w-1$ est blanc.

— **Conclusion :** Lorsque la boucle se termine, $w-1=r$ et donc le tableau T est constitué de trois parties :

  — De 1 à $b-1$ qui est bleu
  — De b à $w-1=r$ qui est blanc
  — De r à n qui est rouge

  Le tableau est trié.

Complexité

$r-w$ prend les valeurs de la suite (u_k) telle que $u_0=n-1$ et pour $k\geq 0$, $u_{k+1}=u_k-1$. Donc $u_k=n-1-k$.

Le test d'arrêt est $u_k<0$ qui arrive quand $k>n-1$. Il faut donc n passages en boucle pour finir cet algorithme. Or le coût d'une boucle est en $\Theta(1)$ donc la complexité de cet algorithme est en $\Theta(n)$.

> Remarque importante
>
> Cet algorithme est **linéaire en temps** ($\Theta(n)$) et **constant en espace** (seulement 3 indices). C'est beaucoup plus efficace qu'un tri général en $O(n\log n)$!

Rangement binaire

Principe

Il s'agit d'une variante de l'algorithme du drapeau hollandais mais avec deux couleurs **Noir** et **Blanc**, et donc uniquement deux indices :

— w qui indique la fin du Blanc +1
— b qui indique le début du Noir -1

L'indice de parcours sera w.

> Structure du tableau
>
> ```
> [Blanc | ? (non traité) | Noir]
>   ^        ^               ^
>   1        w               b
> ```
>
> **Il n'y a que deux cas à gérer :**

— **Cas 1 : `T[w] = Blanc`** → Incrémenter w (déjà bien placé)
— **Cas 2 : `T[w] = Noir`** → Échanger `T[w]` avec `T[b]`, puis décrémenter b

Exemple

Soit le tableau initial : `[N, B, N, B, B, N, N, B]`
Déroulement : - $w = 1$, $b = 8$ - `T[1] = N` → Échanger avec `T[8]` → `[B, B, N, B, B, N, N, N]`, $b = 7$ - `T[1] = B` → $w = 2$ - `T[2] = B` → $w = 3$ - `T[3] = N` → Échanger avec `T[7]` → `[B, B, N, B, B, N, N, N]`, $b = 6$ - ...
Résultat final : `[B, B, B, B, N, N, N, N]`

Pseudo-code

```
ALGORITHME BlancetNoir(T):
DONNEES
  T : tableau de deux couleurs Noir et Blanc de taille n
VARIABLES
  b, w : entiers
DEBUT
  w ← 1
  b ← n
  TQ w   b FAIRE
    SI T[w] = Blanc ALORS
      w ← w + 1
    SINON
      Echanger(T, w, b)
      b ← b - 1
    FSI
  FTQ
FIN
```

Arrêt

De la même façon que pour l'algorithme du drapeau hollandais, le test d'arrêt dépend de $b - w$ qui prend les valeurs d'une suite strictement décroissante et donc valide $b - w < 0$ à un moment donné.

Validité

De la même façon, l'invariant de boucle est : "le sous-tableau de T de 1 à $w-1$ est Blanc et de $b+1$ à n est Noir."

On le montre de la même façon que pour le drapeau hollandais.

Complexité

De la même façon, la complexité est en $\Theta(n)$.

Comparaison et applications

Tableau comparatif

Algorithme	Nombre de couleurs	Complexité	Espace	Caractéristique
Drapeau hollandais	3	$\Theta(n)$	$O(1)$	3 indices, 3 zones
Rangement binaire	2	$\Theta(n)$	$O(1)$	2 indices, 2 zones

Applications pratiques

Ces algorithmes de partitionnement sont utilisés dans :

- **Tri rapide (QuickSort)** : le partitionnement est au cœur de l'algorithme
- **Algorithmes de sélection** : trouver le k-ième plus petit élément
- **Traitement d'images** : segmentation par couleurs
- **Filtrage de données** : séparer des données selon un critère binaire ou ternaire

! Points clés

- Ces algorithmes sont **en place** (pas de tableau auxiliaire)
- Ils sont **linéaires** en temps ($\Theta(n)$)
- Ils maintiennent un **invariant de boucle** qui garantit la correction
- Le principe de **partitionnement** est fondamental en algorithmique

Généralisation

Le principe peut se généraliser à k couleurs :

- Utiliser k indices pour délimiter les zones
- Complexité reste en $O(n)$ avec $O(k)$ indices
- Plus complexe à implémenter mais même principe d'invariant