

EA4 – Éléments d’algorithmique

TD n° 6 : encore des tris...

Exercice 1 : tri drapeau

Le problème du *drapeau hollandais* est le suivant : le tableau à trier contient trois types d’éléments, les bleus, les blancs et les rouges et on souhaite les trier par couleur. Tous les éléments d’une même couleur ne sont pas identiques, il ne suffit donc pas de les compter.

1. On dit qu’un algorithme de tri est *stable* s’il préserve l’ordre du tableau initial pour les éléments de même couleur. Proposer un algorithme linéaire et stable (mais pas en place) pour résoudre ce problème.
2. Expliquer en quoi la complexité cet algorithme de tri ne contredit pas la borne $\Omega(n \log n)$ démontrée en cours.
3. Proposer un algorithme linéaire et en place (mais pas stable) dans le cas où il n’y a que deux couleurs (bleu et rouge).
4. Comment adapter l’algorithme précédent au cas où le tableau contient un unique élément blanc, dans sa première case ?
5. Généraliser l’algorithme de la question 3 au cas de trois couleurs. Pour cela, on maintiendra l’invariant suivant :

« le tableau est constitué de 4 segments consécutifs, un bleu, un blanc, un non exploré, et un rouge ».

Utiliser trois curseurs pour représenter les frontières entre ces segments.

Remarque : il n’existe pas à notre connaissance d’algorithme à la fois linéaire, stable et en place pour ce problème ; on peut en revanche modifier légèrement l’algorithme de la question 5 pour obtenir un algorithme stable et en place (mais pas linéaire, donc).

Exercice 2 : tri par base

Le *tri par base* est un algorithme qui trie selon l’ordre lexicographique une liste L de n mots de longueur ℓ sur un alphabet fini A de taille k – par exemple, des nombres entiers écrits en base k . L’algorithme consiste en ℓ étapes, chacune réalisant un tri des données en fonction d’un des chiffres. La première étape concerne le *chiffre le moins significatif* (celui des unités), puis le chiffre des « k -aines », jusqu’au chiffre le plus significatif (de degré $\ell - 1$).

1. Donner un algorithme du tri par base *stable* de complexité linéaire en n . Justifier sa stabilité et sa complexité en temps. Quelle est sa complexité en espace ?
2. Décrire l’exécution du tri par base sur la liste 657, 819, 457, 329, 416, 720, 455.
3. Pourquoi ne pas exécuter les étapes dans l’ordre inverse ?

Exercice 3 : (partiel 2022)

On considère l'algorithme `foo` ci-dessous.

```
def foo(T, deb = 0, fin = None) :
    if fin == None : fin = len(T)
    if fin - deb < 2 : return T
    if fin - deb == 2 :
        if T[deb] > T[deb+1] :
            T[deb], T[deb+1] = T[deb+1], T[deb]
        return T
    un_tiers = (fin - deb) // 3
    b1, b2 = deb + un_tiers, fin - un_tiers
    foo(T, deb, b2)
    foo(T, b1, fin)
    foo(T, deb, b2)
    return T
```

1. Décrire son déroulement pour chacun des deux appels :

- `foo([4, 3, 2, 1])`
- `foo([6, 5, 4, 3, 2, 1])` sans détailler les appels pour $\text{fin} - \text{deb} < 5$

2. Émettre une conjecture \mathcal{C} sur l'état de T après exécution de `foo`.

3. On souhaite démontrer \mathcal{C} . Soit P_n la propriété « \mathcal{C} est vraie pour tout tableau de taille au plus n ». Soit $n > 2$ un entier tel que P_n est vraie, et considérons un tableau T de taille $n + 1$. Comparer les tailles des trois sous-tableaux $T_0 = T[\text{deb}:b_1]$, $T_1 = T[b_1:b_2]$ et $T_2 = T[b_2:\text{fin}]$. Que peut-on dire de ces trois sous-tableaux après le premier appel `foo(T, deb, b2)` ?

Après l'appel `foo(T, b1, fin)` ?

Après le deuxième appel `foo(T, deb, b2)` ?

Conclure.

4. Soit $S(n)$ le nombre de comparaisons effectuées lors d'un appel à `foo` sur un tableau de taille n . Donner une définition récursive de $S(n)$.

5. Comparer la complexité de `foo` à celle des algorithmes classiques résolvant le même problème.

Question début du cours

$$A(n) = \begin{cases} 0 & \text{si } n \leq 3 \\ A(n-3) + n & \text{autre cas} \end{cases}$$

$$n = 3m$$

$$m = \frac{n}{3}$$

$$\begin{aligned} A(3m) &= A(3(m-1)) + 3m \\ &= A(3(m-2)) + 3(m-1) + 3m \\ &= A(3(m-2)) + 3(m-1) + 3m \\ &= A(0) + 3 + 3 \times 2 + \dots + 3(m-1) + 3m \\ &= 3(1+2+3+\dots+m) \end{aligned}$$

$$A(n) = A(n-4) + 10$$

$$\begin{aligned} A(n) &= A(n-4) + 10 \\ &= A(n-8) + 10 + 10 \end{aligned}$$

⋮
⋮

$$\begin{aligned} &= A(0) + \underbrace{10 + \dots + 10}_{\frac{n}{4} \text{ fois}} \end{aligned}$$

$$n = 4m \Rightarrow m = \frac{n}{4}$$

$$A(4m) = A(\underbrace{4m - 4}_{4(m-1)})$$

$$AC(n) = A\left(\frac{n}{3}\right) + 12$$

$$n = 3^k \Leftrightarrow k = \log_3(n)$$

$$\begin{aligned} AC(3^k) &= AC(3^{k-1}) + 12 \\ &= AC(3^{k-2}) + 12 + 12 \\ &= AC(3^0) + 12 + \dots + 12 \end{aligned}$$

"Counting sort"

T taille n

numbers entre 1 et 5

$$C = [\underbrace{0}_1, \underbrace{0}_1, \underbrace{0}_1, \underbrace{0}_1, \underbrace{0}_1]$$

↑
nb1 nb2 5

$$T = [1, \dots, 1] + [2, \dots, 2] + [5, \dots, 5]$$

Complexité $\Theta(n)$

Tri par comptage

$$T = [\underbrace{1, \dots, 1, 1}_1, \underbrace{2, 2, \dots, 2}_1, \underbrace{3, \dots, 3}_1, \underbrace{4, \dots, 4}_1, \underbrace{5, \dots, 5}_1]$$

12

3

5

4

9

$$T = [3, 1, 5, 1, 2, 2, 1, 5]$$

for x in T

$$C[x-1]_+ = 1$$

$$T = []$$

for i in range [5]

$$T += [i + 1] + C[i]$$

Stabilité

$$[0, 0, 0, 0, 0]$$

$$T [(2,0), (1,5), (2,3), (5,0)]$$

$$[0, 0, 0, 0, 0]$$

$$T [(1,5), (2,0), (2,3), (5,0)]$$

$$i \equiv 0 \pmod{3}$$

$$(2,3), (2,0)$$

$$1 \pmod{3}$$

$$2 \pmod{3}$$

En place : N'allowe pas de mémoire en plus

Ex1:

$$\text{ly } T = [1, 3, 2, 1, 5, 4, 5, 6]$$

$$B, W, R = [], [], []$$

for x in T :

if $\text{color}(x) = \text{white}$
 $W \text{ append}(x)$

$$B[2, 5, 6]$$

$$W[3, 1, 4]$$

if $\text{color}(x) = \text{blue}$
 $B \text{ append}(x)$

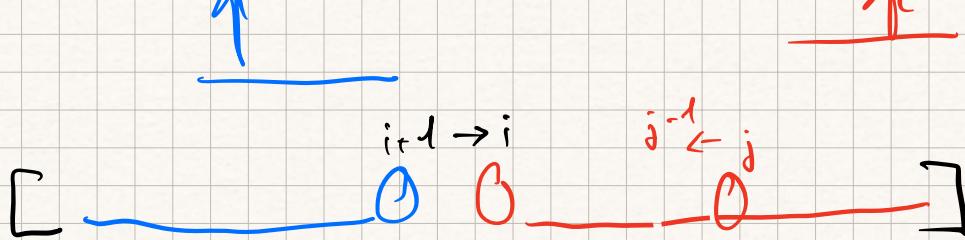
$$R[1, 5]$$

if $\text{color}(x) = \text{red}$
 $R \text{ append}(x)$

$$B + W + R$$

return $B + W + R$

$$\text{ly } T[1, 3, 2, 4, 5, 6, 5, 2, 1, 2]$$



$$i = 0, j = n-1$$

Tant que $i < j$

si $\text{color}_{i+1}(T[i]) = \text{blue}$

swapping:

$$T[i], T[j] = T[j], T[i]$$
$$j - 1 = 1$$

Si $i = j$ l'algorithme s'arrête

$$\Leftrightarrow j - i = 0$$

Au début, $j - i = n - 1 - 0 = n - 1$

A chaque itération, $j - i$ diminue de 1
Donc l'algorithme fait $n - 1$ itérations

$$\text{tq } i = 1, j = n - 1$$

Tant que $i < j$

si $\text{color}_{i+1}(T[i]) = \text{blue}$

Alors : Si $\text{color}(T[i]) = \text{orange}$

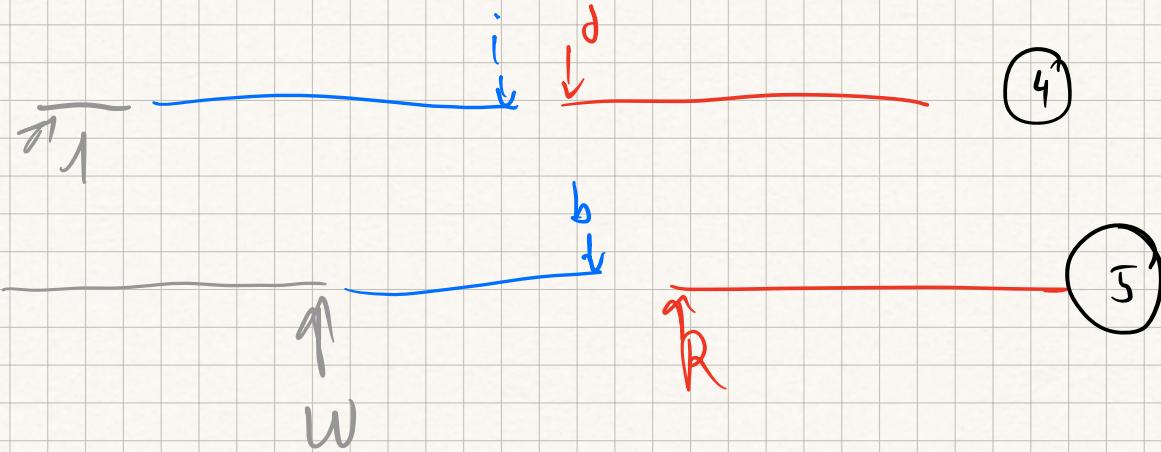
$$T[i], T[j] = T[j], T[i]$$

$$j - = 1$$

Sinon :

$$T[1], T[i] = T[i], T[1]$$

5



$$W = \emptyset, b = 0, R = n - 1$$

Tant que ($b \leq R$)

if ($T[b] \leq b$ lancer) {

$T[b], T[w] = T[b], T[w]$

$w += 1$, continue

if ($T[b] == \text{large}$) {

$T[b], T[R] = T[R]. T[b]$

$R -= 1$, continue

$b += 1$

}

Ex 1

$\text{lyr foo}([4, 3, 2, 1])$

$$b_1 = \text{deb} + \frac{n}{3} \rightarrow \frac{n}{3}$$

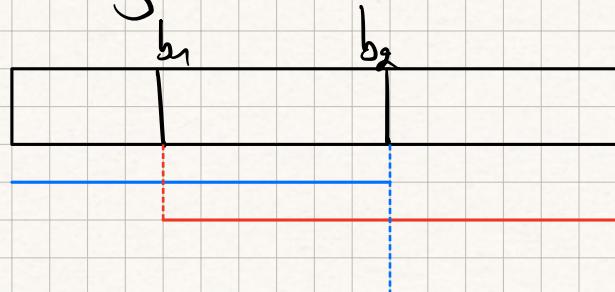
$$b_2 = \text{fin} - \frac{n}{3} \rightarrow \frac{2n}{3}$$

$T[\text{deb}, b_2]$

$T[b_1, \text{fin}]$

$T[\text{deb}, b_2]$

$$m = \frac{n}{3}$$



4, 3, 2

3, 1 2

3, 2, 1

2, 3, 4

$$S(n) = \begin{cases} 0 & \text{si } n \leq 1 \\ 1 & \text{si } n = 2 \\ 33\left(\frac{2}{3}n\right) + n^0 & \text{otherwise} \end{cases}$$

≈ 271

$$a = 3$$

$$b = \frac{3}{2}$$

$$S(n) = \Theta\left(n^{\log_{\frac{3}{2}} 3}\right)$$

#

nbr de complexité