Université de Paris

L2 informatique - Année 2021–2022

TD d'Éléments d'Algorithmique n° 2

Dans ce TD, “trié” signifie "trié par ordre croissant".
*** Les exercices marqués d'une étoile sont à faire à la maison.**

Exercice 1. *Tri sélection.*
Exécutez à la main le tri par sélection vu en cours sur le tableau suivant :

0	3	1	2

Combien de comparaisons avez-vous dû faire ?

Exercice 2. *Tri sur tri.*
Dans cet exercice on utilise

- la fonction triSelection du cours, qui trie les tableaux dont les éléménts sont comparés par l'opérateur $<$, et
- la fonction inf du TD 1, qui prend en argument deux mots et qui retourne -1 si le premier mot est avant le second dans l'ordre lexicographique, 0 si les deux mots sont les mêmes, et +1 sinon.

Écrire une fonction qui prend en argument un tableau de tableaux de caractères `T`, et qui :

- trie par selection chacun des éléménts de `T`, pour l'ordre alphabétique (pour comparer deux caractères on utilise l'opérateur $<$).
- trie ensuite `T` par selection, pour l'ordre lexicographique.

Par exemple le tableau

c	a	r	a	r	b	r	e	c	e	c	i

est transformé en

a	b	e	r	r	a	c	r	c	c	e	i

Exercice 3. *Tri bourrin.*

Algorithm 1 Tri bourrin

Entrée : tableau T

```
1: fonction triBourrin(T)
2:     n ← longueur de T
3:     pour i ← 0 à n − 2 faire
4:         pour j ← i + 1 à n − 1 faire
5:             si T[i]>T[j] alors
6:                 échanger T[i] et T[j]
```

1. Cet algorithme vous semble-t-il correct pour trier un tableau de taille `n` ? À quel algorithme du cours pourriez vous le comparer ?
2. Que dire de son nombre de comparaisons ? son nombre d'affectations ?

Université de Paris

L2 informatique - Année 2021–2022

TD d'Éléments d'Algorithmique n° 2

Dans ce TD, “trié” signifie "trié par ordre croissant”.
*** Les exercices marqués d'une étoile sont à faire à la maison.**
Exercice 1. *Tri sélection.*
Exécutez à la main le tri par sélection vu en cours sur le tableau suivant :

0	3	1	2

Combien de comparaisons avez-vous dû faire ?

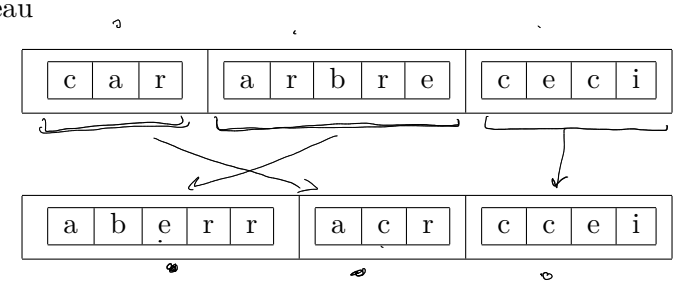
Exercice 2. *Tri sur tri.*
Dans cet exercice on utilise

- la fonction triSelection du cours, qui trie les tableaux dont les éléménts sont comparés par l'opérateur $<$, et
- la fonction inf du TD 1, qui prend en argument deux mots et qui retourne -1 si le premier mot est avant le second dans l'ordre lexicographique, 0 si les deux mots sont les mêmes, et +1 sinon.

Écrire une fonction qui prend en argument un tableau de tableaux de caractères `T`, et qui :

- trie par selection chacun des éléménts de `T`, pour l'ordre alphabétique (pour comparer deux caractères on utilise l'opérateur $<$).
- trie ensuite `T` par selection, pour l'ordre lexicographique.

Par exemple le tableau



est transformé en

Exercice 3. *Tri bourrin.*

Algorithm 1 Tri bourrin

Entrée : tableau T

```
1: fonction triBourrin(T)
2:     n ← longueur de T
3:     pour i ← 0 à n − 2 faire
4:         pour j ← i + 1 à n − 1 faire
5:             si T[i]>T[j] alors
6:                 échanger T[i] et T[j]
```

1. Cet algorithme vous semble-t-il correct pour trier un tableau de taille `n` ? À quel algorithme du cours pourriez vous le comparer ?
2. Que dire de son nombre de comparaisons ? son nombre d'affectations ?

Algorithm 2 Tri du drapeau hollandais

Entrée : tableau T contenant uniquement les valeurs 0, 1 et 2

```
1:  fonction triDrapeau(T)
2:      p ← 0
3:      m ← 0
4:      g ← longueur(T) − 1
5:      tant que m ≤ g faire
6:          switch T[m] faire
7:              case 0
8:                  échanger T[m] et T[p]
9:                  m ← m + 1
10:                 p ← p + 1
11:             case 1
12:                 m ← m + 1
13:             case 2
14:                 échanger T[m] et T[g]
15:                 g ← g − 1
```

Exercice 4. *Problème du drapeau hollandais.*
L'algorithme ci-dessus permet de trier un tableau contenant uniquement trois valeurs différentes. Ici, on suppose que ces valeurs soient les entiers 0, 1 et 2.

1*. Implémentez triDrapeau en Java et déposez votre code sur Moodle.

2. Soit `T`= | 1 | 1 | 0 | 2 | 0 | 2 |; évaluez l'appel triDrapeau `T` à la main.

3. Qu'est-ce qui garantit que tout appel de triDrapeau termine ?

4. Montrer que pendant l'exécution de triDrapeau, les propriétés suivantes sont satisfaites :
   (a) $p \leq m$
   (b) les éléments d'indice inférieur à p sont des 0.
   (c) les éléments d'indice supérieur à g sont des 2.
   (d) les éléments d'indice de p à $m - 1$ sont des 1.

5. Conclure qu'à la fin de l'exécution de l'algorithme, le tableau est trié.

6. Combien d'echanges sont effectués pendant l'exécution de triDrapeau, dans le pire des cas, la taille de `T` étant n ? Et dans le meilleur de cas ?

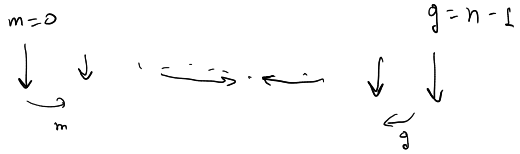
7. Même question que ci-dessus, pour les comparaisons.

Exercice 5. *Drapeau polonais*.*
On veut adapter l'algorithme du tri drapeau au cas simple où le tableau à trier ne contient que deux valeurs différentes (par exemple 0 et 1). Ecrivez la fonction `triDrapeauBicolore`, en vous inspirant du code de `triDrapeau`.

Algorithm 2 Tri du drapeau hollandais

Entrée : tableau T contenant uniquement les valeurs 0, 1 et 2

```
1:  fonction triDrapeau(T)
2:      p ← 0
3:      m ← 0
4:      g ← longueur(T) − 1
5:      tant que m ≤ g faire
6:          switch T[m] faire
7:              case 0
8:                  échanger T[m] et T[p]
9:                  m ← m + 1
10:                 p ← p + 1
11:             case 1
12:                 m ← m + 1
13:             case 2
14:                 échanger T[m] et T[g]
15:                 g ← g − 1
```



Exercice 4. *Problème du drapeau hollandais.*
L'algorithme ci-dessus permet de trier un tableau contenant uniquement trois valeurs différentes. Ici, on suppose que ces valeurs soient les entiers 0, 1 et 2.

1*. Implémentez triDrapeau en Java et déposez votre code sur Moodle.

2. Soit `T`= | 1 | 1 | 0 | 2 | 0 | 2 |; évaluez l'appel triDrapeau `T` à la main.
3. Qu'est-ce qui garantit que tout appel de triDrapeau termine ?
4. Montrer que pendant l'exécution de triDrapeau, les propriétés suivantes sont satisfaites :
   (a) $p \leq m$
   (b) les éléments d'indice inférieur à p sont des 0.
   (c) les éléments d'indice supérieur à g sont des 2.
   (d) les éléments d'indice de p à $m-1$ sont des 1.
5. Conclure qu'à la fin de l'exécution de l'algorithme, le tableau est trié.
6. Combien d'echanges sont effectués pendant l'exécution de triDrapeau, dans le pire des cas, la taille de `T` étant n ? Et dans le meilleur de cas ?
7. Même question que ci-dessus, pour les comparaisons.

Exercice 5. *Drapeau polonais*.*
On veut adapter l'algorithme du tri drapeau au cas simple où le tableau à trier ne contient que deux valeurs différentes (par exemple 0 et 1). Ecrivez la fonction `triDrapeauBicolore`, en vous inspirant du code de `triDrapeau`.

Exo 2 : Supposons qu'on a déjà deux fonctions :

- char[] triSelection(char[])
- inf inf(char[], char[]) ← trouve le min entre deux mots.

Entrée : tableau T[][] de char

```
fonction triSurTri(T) :
    n ← taille de T
    pour i ← 0 à i ← n-1
        T[i] = triSelection(T[i]);

    pour i ← 0 à n-2
        min = i;
        pour j ← i+1 à n-1
            si inf(T[min], T[j]) == 1
                echange T[min] ↔ T[j]
                min = j;
```

Exo 3 :

$i = [0 \dots n-2]$

$j = [i+1 \dots n-1]$

1 comparaison

$$\left.\begin{array}{l} n-1 \\ + \; n-2 \\ + \; \vdots \\ + \; 1 \end{array}\right\} \frac{n(n-1)}{2} \text{ comparaisons} \qquad O(n^2)$$

$$\sum_{i=1}^{n-1} i$$

$$1+2+3+4 = (10) = \frac{5 \cdot 4}{2}$$

$$1+2+3+\dots+n-1 = \frac{n(n-1)}{2}$$

Affectations : $1 \rightsquigarrow$ n ← longeur $\qquad O(n^2)$

$0 \dots n-2$ $\quad n-1 \rightsquigarrow$ pour la variable i

$\frac{n(n-1)}{2} \rightsquigarrow$ pour la variable j

meilleur : 0 $\rightsquigarrow$ pour echange (il y a 3 affectations par echange effectué)

pire : $3 \cdot \frac{n(n-1)}{2}$

1 ... n-1
2 ... n-2
3 ... n-1
n-1

Exo 4 :

2. [0 0 1 1 2 2] ←
   p g m

---

3. while loop peut ne Jamais terminer !

```
while (true) {
    i = i + 1;
}
```

---

pour chaque iterations :

- soit g diminue
- soit m agrandit

Vu que g commence à n-1 et m commence à 0, à moment donné m va depasser g.

---

4. Par recurence

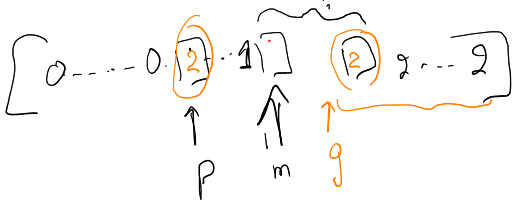
Cas de base :

a) p=0, m=0 alors p≤m ✓
b) il n'y a pas d'elements ✓
c) —— // —— ✓
d) —— // ——

Cas inductif :

(a) p s'increment seulement si m s'increment (cas 0)
dans aucun cas m ne decrement !

(b, c, d) Disons que c'est vrai pour K repetitions.
pour K+1.

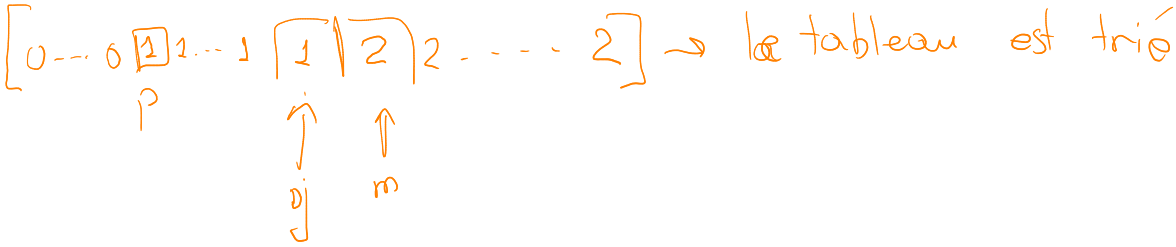


b) Si on est dans cas 1, 2, les variables d'indice < p ne changent pas. ✓
si on est dans cas 0 :
On echange T[m]=0 avec T[p] et p ← p+1 ✓

c) Cas 0, 1 ✓

Cas 2 : pareil avec (b)

5. Quand $m > g$:



→ le tableau est trié