EA CMTD 2 24.09.2020

Tri de tableau par sélection

① On parcourt le tableau t de gauche à droite

② Au moment où on va analiser l'élément $t[i]$ du tableau, tous les éléments qui sont à gauche sont déjà triés.

③ On va trouver le plus petit élément à droite de $t[i]$, c'est-à-dire, le plus petit élément min du tableau qui n'est pas encore trié.

④ On échange min et $t[i]$.

exemple :

6	3	1	4	2

$i = 0$

1	3	6	4	2

$i = 1$

1	2	6	4	3

$i = 2$

1	2	3	4	6

$i = 3$

1	2	3	4	6

ALGO

```
TriSelection (tableau T) :
    n ← longueur (T)
    pour i de 0 à n-2
        min ← i    // l'index où se trouve le minimum
        pour j de i+1 à n-1
            si T[j] < min
                min ← j
        échanger T[i] et T[min]
```

$i = 0$ $n-1$ fois
$i = 1$ $n-2$ fois
...
$i = n-2$ 1 fois

↳ Sur tout le prgm :

$$\sum_{i=0}^{n} i = \frac{n(n-1)}{2} = \frac{n^2 - n}{2}$$

Preuve que l'algorithme est correct :

Invariant de boucle : $\forall i$ de 0 à $n-2$

① Le tableau est une permutation du tableau initial

② Les $i-1$ premiers éléments du tableau sont triés.

③ ———————— sont $\leq$ aux restants.

Montrons l'invariant

Init : $i = 0$ ⇒ c'est vrai

Récurrence : Supposons qu'il est vrai pour i. Montrons le pour $i+1$.

① est vraie (on ne fait que des échanges).

② Je me sers des hypothèses d'induction (ou de récurrence) ② et ③ pour le montrer.

③ Par l'hypothèse d'induction ③. ④ je rajoute le minimum

TD 2

Exercice 1

0 3 1 2
0 3 1 2
0 1 3 2
0 1 2 3

} 6 comparaisons

Exercice 2

```
void triChar(char[][] t){
    for (int i = 0; i < t.length; i++){
        triSelection(t[i]);
    }
    for (int i = 0; i < t.length; i++){
        int min = i;
        for (int j = i+1; j < t.length; j++){
            if (inf(t[min], t[j]) == 1) min = j;
        }
        char[] tmp = t[i];
        t[i] = t[min];
        t[min] = tmp;
    }
}
```

Exercice 3

1) Oui. Ça ressemble au tri par select° (en + efficace).

2) comparaisons : $\frac{n(n-1)}{2}$

affectations : $1 + 3\frac{n(n-1)}{2}$ (échanger 2 variables prend 3 affectations)

Exercice 4

1)

```
void triDrapeau(int[] t){

    int p = 0;
    int m = 0;
    int g = t.length - 1;

    while (m >= g){

        switch (t[m]){

            case 0 : int tmp = t[m];
                     t[m] = t[p];
                     t[p] = tmp;
                     m += 1;
                     p += 1;
                     break;

            case 1 : m += 1;
                     break;

            case 2 : int tmp = t[m];
                     t[m] = t[g];
                     t[g] = tmp;
                     g -= 1;
        }
    }
}
```

2)

$m : 0 + 1$ $\quad T = 1\ 1\ 0\ 2\ 0\ 2$
$p : 0$
$g : 6 - 1 = 5$

$m = 1 + 1$ $\quad T = 1\ 1\ 0\ 2\ 0\ 2$
$p = 0$
$g = 5$

$m = 2 + 1$ $\quad T = 1\ 1\ 0\ 2\ 0\ 2$
$p = 0 + 1$
$g = 5$

$m = 3$ $\quad T = 0\ 1\ 1\ 2\ 0\ 2$
$p = 1$
$g = 5 - 1$

$m = 3$ $\quad T = 0\ 1\ 1\ 2\ 0\ 2$
$p = 1$
$g = 4 - 1$

$m = 3 + 1$ $\quad T = 0\ 1\ 1\ 0\ 2\ 2$
$p = 1 + 1$
$g = 3$

$m = 4$ $\quad T = 0\ 0\ 1\ 1\ 2\ 2$
$p = 2$
$g = 3$ $\quad m > g$ FIN

3) À chaque tour, soit $n \nearrow$ soit $g \searrow$. On ne peut pas incrémenter g. décrémenter n.

Donc à un moment $n > g$.

4) (a) <u>Init :</u> $p = n = 0$

<u>Récurrence :</u> On suppose $p \leq n$ avant la boucle.
Montrons que ça reste vrai après une itération.

Soit $T[n] = 0$ Alors $p = p+1$, $n = n+1$ On a tjs $p \leq n$

Soit $T[n] = 1$ Alors $n = n+1$ On a $p \leq n \leq n+1$

Soit $T[n] = 2$ On garde $p \leq n$

<u>Conclusion :</u> On a tjs $p \leq n$.

(b) <u>Init</u> $p = 0$ Pas d'élément d'indice < 0.

<u>Récurrence :</u> On suppose la prop vraie avant la boucle.
Montrons qu'elle reste vraie après une itération.

Soit $t[n] = 2$ ou 1
Comme $p \leq n$, les éléments d'indice $< p$ ne changent pas.

Soit $t[n] = 0$
Alors $t[p] = 0$ et on incrémente p.
Comme tous les éléments d'indices $< p-1$ sont 0 (par HR)
$\Rightarrow$ tous les éléments $< p$ sont 0.

<u>Conclusion :</u> OK

(c) <u>Init</u> : $g = n - 1$: pas d'éléments $t(i)$ tq $i > g$.

<u>Récurrence :</u> On suppose la prop vraie avant la boucle. Montrons qu'elle reste vraie après une itération.

Soit $t[n] = 0$ ou 1 : rien ne change

Soit $t[n] = 2$
Alors on met 2 dans $t(g)$
Puis on décrémente g

Par HR, tous les éléments $t(i)$ tq $i > g+1$ sont égaux à 2.
De plus $t[g+1] = 2$

Donc tous les éléments $t(i)$ tq $i > g$ sont égaux à 2.

(d) <u>Init</u> : pas d'élément entre p et $m-1$

<u>Récurrence</u> :

Soit $t[m] = 2$
Comme $m \leq g$, rien ne change entre p et $m-1$.

Soit $t[m] = 1$
Alors on incrémente m.
Par HR : tous les $t[i]$ tq $p \leq i \leq m-2$ sont égaux à 1.
De plus $t[m-1] = 1$.
Donc tous les éléments $t[i]$ tq $p \leq i \leq m-1$ sont égaux à 1.

Soit $t[m] = 0$
Par HR, $t[p] = 1$ et $\forall i$ $p+1 \leq i \leq m-1$ $t[i] = 1$
Quand on échange $t[m]$ et $t[p]$, on a :

$$\forall i \text{ tq } p+1 \leq i \leq m \quad t[i] = 1$$

On incrémente p et décrémente m de 1.

Donc on a bien $\forall i$ tq $p \leq i \leq m-1$ $t[i] = 1$

5) Les éléments à trier sont ceux dont les indices sont entre m et g (compris).

À la fin de l'exécution $m > g$ donc tous les éléments ont été placés :

soit	$i < p$	$t[i] = 0$
soit	$p \leq i \leq m-1$	$t[i] = 1$
soit	$g < m \leq i$	$t[i] = 2$

Comme $p \leq m \leq g$, le tableau est bien trié.

6) Dans le <u>meilleur</u> des cas : (nombre de 0) échanges $\qquad$ 0 0 ... 1 1 ... 2 2 ... (FIN)

Dans le <u>pire</u> des cas : (nombre de 0) + (nombre de 2) échanges $\qquad$ 2 2 ... 0 0 ... 1 1 ...
puis 1 1 ... 0 0 ... 2 2 ... (FIN)

7) Soient a = nb de 0 ; b = nb de 1 ; c = nb de 2.

Dans le <u>meilleur</u> des cas : $a\,n + b \cdot 2n$ (on s'arrête avant de tester les 2)

Dans le <u>pire</u> des cas : $a\,n + b \cdot 2n + c \cdot 3n$ (on teste chaque valeur)

Exercice 5

```
void triDrapeauBicolore (int[] t){
    int p = 0;
    for (int i = 0; i < t.length; i++){
        if (t[i] == 0){
            t[i] = 1;
            t[p] = 0;
            p += 1;
        }
    }
}
```