TD 2

× Terminaison d'un algorithme
× Validité d'un algorithme

Exercice 1

1). La fonction contient des opérations élémentaires et une boucle while
. la boucle :
- contient des opérations élémentaires
- est répétée n fois // i parcourt un nombre fini de valeurs

Donc la fonction termine.

Aussi : × i est incrémenté à chaque tour de boucle
× on sort de la boucle si i dépasse n.

Q2

Brouillon écrire les 1ères valeurs de tmp_i (à dispenser pour trouver la formule)

$tmp_2 = 1$
$tmp_2 = tmp_2 \times 1 = 1$
$tmp_3 = tmp_2 \times 2 = 1 \times 2$
$tmp_4 = tmp_3 \times 3 = 1 \times 2 \times 3$
$tmp_5 = tmp_4 \times 4 = 1 \times 2 \times 3 \times 4$

⇝ Conjecture $tmp_i = (i-1)!$.

Mq $tmp_i = (i-1)!$ pour $i \in [\![1, \ldots, n+1]\!]$
récurrence portée sur i

• Base : $i = 1$ $tmp_2 = 1 = 0!$ donc $tmp_2 = 0!$.

- Induction. Soit $i \in \{1, \dots, n\}$, on suppose $tmp_i = (i-1)!$

Mq $tmp_{i+1} = i!$

HR $= (i-1)!$

Par définition de tmp_i : $\boxed{tmp_{i+1} = tmp_i \times i}$

① tmp_{i+1} = calcul après le tour de boucle de i

donc $tmp_i = (i-1)! \times i = i!$

Conclusion $\begin{cases} tmp_2 = 0! \\ \forall i \in \{1, \dots, n\} \; tmp_i = (i-1)! \Rightarrow tmp_{i+1} = i! \end{cases}$

donc par récurrence faible

$\boxed{\forall i \in \{1, \dots, n+1\} \; tmp_i = (i-1)!}$ -

Invariant de boucle : $\forall i \in \{1, \dots, n+1\} \; tmp_i = (i-1)!$

Q3 Mq FactorielleIt(n) renvoie $n!$

Méthode calculer la valeur de l'invariant de boucle à la fin de la boucle et s'en servir pour conclure.

À la fin de la boucle tmp vaut tmp_{n+1}, et $tmp_{n+1} = n!$ d'après Q2.

Donc la fonction renvoie $n!$ donc elle est valide

Exercice 3

Q1

- La fonction contient des opérations élémentaires et une boucle
- La boucle contient des opérations élémentaires est répétée un nombre fini de fois
  car i parcourt un nombre fini de valeurs
  car | i est incrémenté à chaque tour de boucle
  | i commence à 0
  | elem est dans tab
  | on sort de la boucle lorsque tab(i) = elem

Donc la fonction termine

Q2

elem
tab[0, . , i-1]

on est à la ième itération
⇒ la condition du while est vraie $\forall k \in \{0, \dots, i-1\}$

on a appliqué le while à 0, . , i-1
donc la condition du while est vérifiée pour 0, . , i-1.

donc $\forall k \in \{0, \dots, i-1\}$ elem ≠ tab(k)

ie elem n'est pas dans tab[0, ..., i-1]

Invariant de boucle

$\forall i \geq 1$ elem n'appartient pas à tab[0 ... i-1]

On la démontre par récurrence

- **Base** si i = 1, alors on est entré dans la boucle, et donc elem ≠ tab[0]

- **Induction**

On suppose la propriété vraie pour i, c'est-à-dire elem n'appartient pas à tab[0, ..., i-1]

Ng la propriété est vraie pour $i+1$, elem n'appartient pas à $tab[0, \dots, i]$

Pour avoir $i \leftarrow i+1$ il faut que le test de la boucle

elem $\neq tab[i]$ soit vérifié

→ ! On incrémente i dans la boucle

On en déduit que elem n'appartient pas à $tab[0, \dots, i]$

Conclusion : Par récurrence (Base + induction)

$\forall i \geq 1$ elem n'appartient pas à $tab[0 \dots i-1]$

Q3

À la fin de la boucle

→ Soit $i = 0$ et elem = $tab[0]$

→ Soit $i \geq 1$ et

- elem n'appartient pas à $tab[0 \dots i-1]$
- la condition de boucle est fausse pour i

donc elem = $tab[i]$

Donc i est l'indice minimum tel que

$tab[i]$ = elem.

TD2 Exercice 5

```
def Push(tab,k)
    j = 1
    while (j<k)
        if tab[j] < tab[j-1]
            tmp = tab[j-1], tab[j-1] = tab[j], tab[j] = tmp
        j = j+1
        print("j=", j, " tab=", tab)
```

échange des valeurs entre tab[j] et tab[j-1]

Q1 Push(tab, 6) , tab = [8, 7, 12, 5, 15, 4, 3, 9]

j=2 tab_2 = [7, 8, 12, 5, 15, 4, 3, 9]

j=3 tab_3 = [7, 8, 12, 5, 15, 4, 3, 9]

j=4 tab_4 = [7, 8, 5, 12, 15, 4, 3, 9]

j=5 tab_5 = [7, 8, 5, 12, 15, 4, 3, 9]

j=6 tab_6 = [7, 8, 5, 12, 4, 15, 3, 9]

Brouillon

j=1 tab[1] < tab[0]

donc on fait le if

tmp = tab[0] = 8

tab[0] = tab[1] = 7

tab[1] = tmp = 8

Q2 La fonction contient des opérations élémentaires et une boucle while

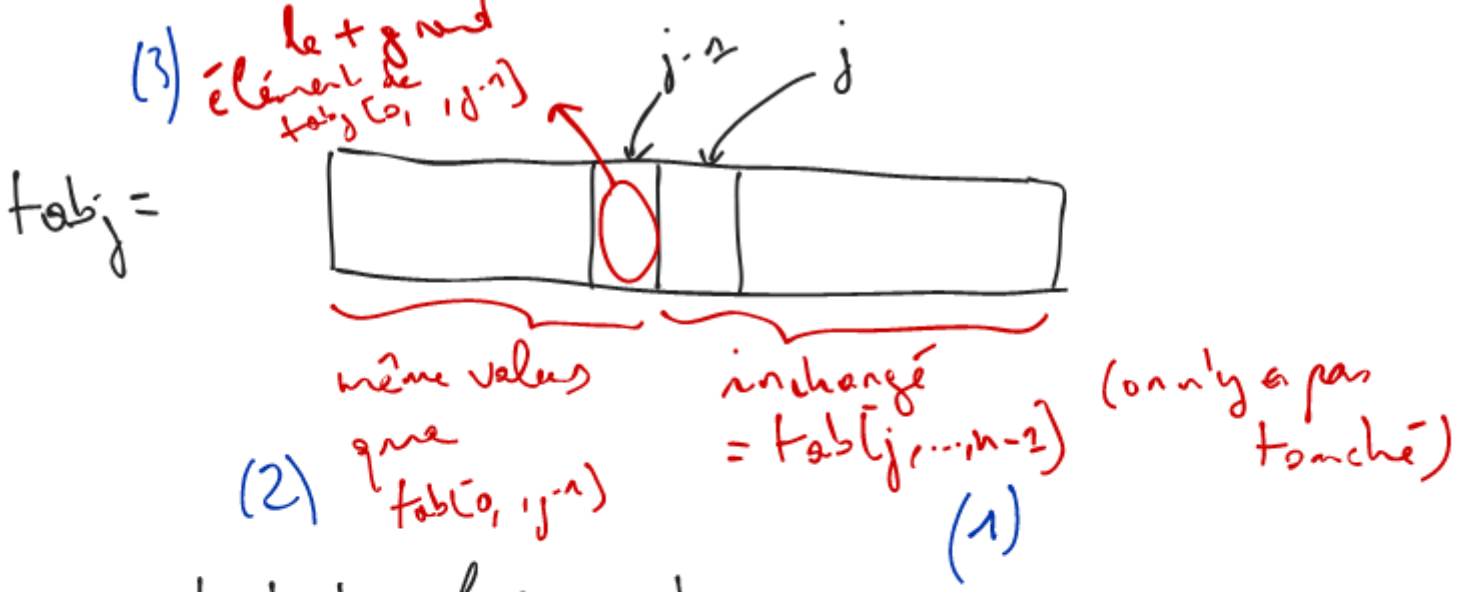
- j est initialisé à 1 et est incrémenté à chaque tour de boucle
- On sort de la boucle lorsque j > k

Donc la fonction termine

Q3 Soit n la taille de tab

Invariant

- $tab_j[j, \dots, n-1] = tab_{j-1}[j, \dots, n-1]$
- $tab_j[j-1]$ contient le plus grand élément de $tab_j[0, \dots, j-1]$ et $tab_j[0, \dots, j-1]$ contient les mêmes valeurs que $tab_{j-1}[0, \dots, j-1]$



Invariant de boucle à montrer

(1) $tab_j[j, \dots, n-1] = tab[j, \dots, n-1]$

(2) $tab_j[0, \dots, j-1]$ contient les mêmes valeurs que $tab[0, \dots, j-1]$

(3) $tab_j[j-1]$ contient le plus grand élément de $tab_j[0, \dots, j-1]$

Démontrons (1), (2) et (3) par récurrence sur $j \in \{1, \dots, k\}$

- **Base** : $j = 1$

(1) $tab_1 = tab$ donc ok

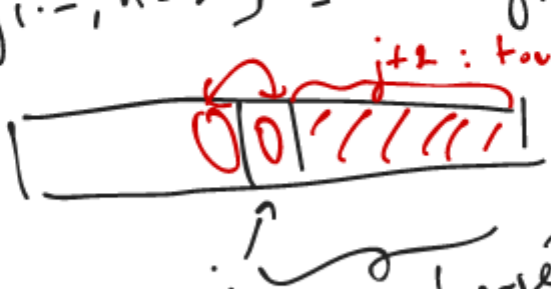
(2) $tab_1 = tab$ donc ok

(3) $tab_1[0, \dots, 1-1] = tab_1[0]$

$tab_1[0]$ contient le plus grand élément de $tab_1[0]$

- **Induction** : Soit $j \in \{1, \dots, k-1\}$. On suppose (1), (2) et (3)

(1) HR : $tab_j[j, \dots, n-1] = tab[j, \dots, n-1]$



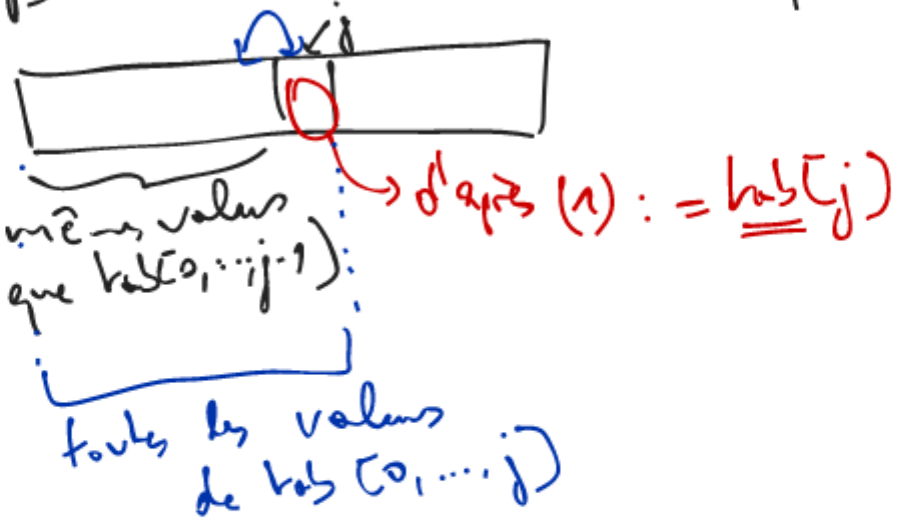
Au rang j, la boucle ne modifie que $tab_j[j]$ et $tab_j[j-1]$

Donc $tab_{j+1}[j+1, \dots, n-1]$ reste inchangé

donc $tab_{j+1}[j+1, \dots, n-1] = tab[j+1, \dots, n-1]$ ✓

(2) HR : $tab_j[0,\dots,j-1]$ mêmes valeurs que $tab[0,\dots,j-1]$

Mq $tab_{j+1}[0,\dots,j]$ contient les mêmes valeurs que $tab[0,\dots,j]$



d'après (1) $tab_j[j] = tab[j]$

donc (1) + HR : $tab_j[0,\dots,j]$ contient toutes les valeurs de $tab[0,\dots,j-1]$

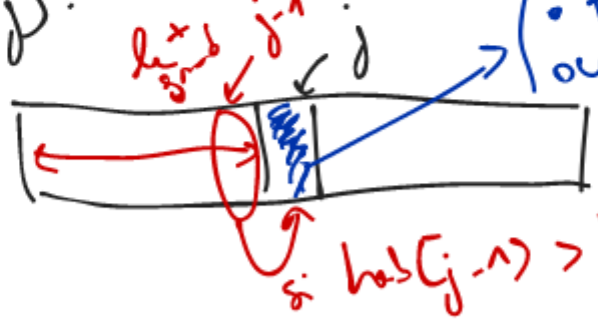
Donc, après l'échange éventuel de $tab_j[j]$ et $tab_j[j-1]$ pour arriver à tab_{j+1}, les valeurs entre 0 et j restent les mêmes

Donc $tab_{j+1}[0,\dots,j]$ contient toutes les valeurs de $tab[0,\dots,j]$.

(3) On suppose que $tab_j[j-1]$ contient le plus grand élément de $tab_j[0,\dots,j-1]$

Mq $tab_{j+1}[j]$ contient le plus grand élément de $tab[0,\dots,j]$.

tab_j =



• $tab_j[j]$ si pas d'échange
ou • $tab_j[j-1]$ si échange

• Si $tab_{j+1}[j] = tab_j[j]$ (*)

alors il n'y a pas eu d'échange donc $tab_j[j] > tab_j[j-1]$ (**)

HR+(**) : $tab_i[j]$ contient le plus grand élément de $tab_j[0,\dots,j]$

et avec (*)+(1) : $tab_{j+1}[j]$ contient le plus grand élément de $tab_{j+1}[0,\dots,j]$

Conclusion : (1), (2) et (3) sont initialisées et héréditaires donc vrai $\forall j \in \{1,\dots,k\}$

Q4 Méthode pour la validité : appliquer l'invariant de boucle à $j=k$ (sortie de boucle)

1. Invariant (3) appliqué à $j=k$
2. Invariant (2) appliqué à $j=k$.

"En sortie de boucle, $j=k$ donc les invariants pour $j=k$ sont vérifiés".