TD 4

Exercice 1

Q1 Complexité en nombre d'additions
= Θ ou O du nombre d'additions

→ S'il y a un meilleur et un pire cas, on donne O(complexité du pire cas)

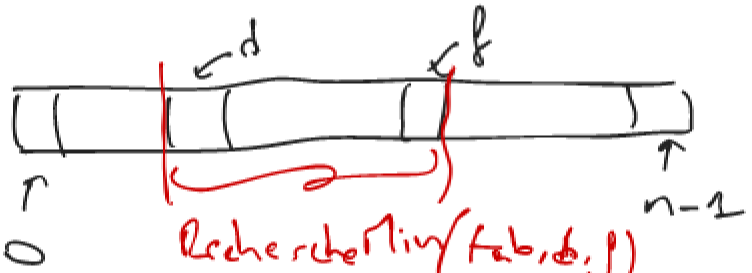
→ Sinon, on donne un Θ

Quel que soit le tableau d'entrée de taille n, il y a une addition à chaque tour de boucle, et il y a n tours de boucle, donc il y a n additions.

Donc la complexité est en $\Theta(n)$.

Q2

tab :



RechercheMin(tab,d,f) renvoie le minimum de ce sous-tableau

- Le corps de boucle comprend 1 test et 2 instructions élémentaires.
- La boucle est exécutée $f-d$ fois

Donc la complexité est $\Theta(f-d)$.

Q3 Méthode :
1. Complexité du corps de boucle
2. Somme de cette complexité sur toutes les itérations

1 Soit c_i la complexité du tour de boucle i

$$c_i = \Theta(n-1-i)$$ car | Appel récursif en $\Theta(n-1-i)$
Reste de la boucle opérations élémentaires en temps constant

2 Donc la complexité de la fonction est

$$\Theta\left(\sum_{i=0}^{n-1} c_i\right) = \Theta\left(\sum_{i=0}^{n-1} (n-1-i)\right)$$

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

$\rightarrow k = n-1-i$

$$\boxed{\sum_{k=0}^{n} k = \frac{n(n+1)}{2}}$$

Donc la complexité est en $\Theta\left(\frac{n(n-1)}{2}\right)$

à simplifier

→ pas de constante
→ pas de terme dominé par un autre

$$\frac{n(n-1)}{2} = \frac{n^2}{\cancel{2}} \cancel{-n}$$ → donc $\boxed{\text{complexité en } \Theta(n^2)}$

Bilan de méthode : Complexité d'une boucle

$$= \sum_{\text{itération}} (\text{complexité de l'itération})$$

En pratique → On calcule la complexité d'une itération
→ On fait la somme

Dans notre cas :

$$c_i = \Theta(n-1-i) + \underbrace{\overbrace{k}^{=\Theta(1)}}_{\text{au plus 5 opérations élémentaires dans le corps de boucle}} \quad \text{avec } 1 \leq k \leq 5$$

$$= \Theta(n-1-i)$$

Exercice 2

Méthode complexité fonction récursive :

→ Complexité des cas de base

→ Etablir une relation de récurrence sur la complexité

$\boxed{Q1}$

Soit u_n le nombre de multiplication effectuées par l'appel fact(n).

Calculons u_n.

→ $u_0 = 0$

→ Si $n \geq 1$, $u_n = 1 + u_{n-1}$

Donc, par substitution, $u_n = 1 + u_{n-1}$

$$= 1 + 1 + u_{n-2}$$

$$= 1 + 1 + 1 + u_{n-3}$$

$$= \ldots$$

$$= \underbrace{1 + \ldots + 1}_{n} + \underbrace{u_{n-n}}_{u_0 = 0}$$

donc $\forall n \geq 0$ $u_n = n$

Donc $\boxed{u_n = \Theta(n)}$

Q2

Tab.

RechercheMinRec(T,d,f) renvoie l'indice de l'élément minimum du ce sous-tableau

Soit c_m le nombre de comparaisons effectuées par RechercheMinRec(T, d, f) lorsque $m = f-d$

→ $c_0 = 1$ $(d == f)$

→ Si $m \geq 1$, $c_m = \underbrace{1}_{1^{er}\ if} + \underbrace{c_{m-1}}_{\text{appel récursif}} + \underbrace{1}_{2^{e}\ if}$

Par substitution, $c_m = 2 + c_{m-1}$

$$= 2 + 2 + c_{m-2}$$

$$= \dots$$

$$= 2m + c_0 = 2m+1$$

Donc $c_m = \Theta(m)$

Donc le nombre de comparaisons est en $\boxed{\Theta(f-d)}$

Q3

RechercheRec:

Tab :

Appel récursif sur ce sous-tableau

non elim ?

oui → retour ✓

→ Meilleur cas : elem est à la fin du tableau
Pas besoin d'appel récursif
Nombre constant d'opérations
(⚠ quel que soit n et tab
$n=0$ ou tab vide ne correspondent pas)
au meilleur cas

→ Pire cas : elem n'est pas dans le tableau
Chaque itération, hors appel récursif, de la fonction possède un nombre d'instructions borné par une constante k.
Donc, la complexité C_n dans le pire cas pour un tableau de taille n vérifie $C_n = k + C_{n-1}$
et dans le pire cas, il y a n itérations (on va jusqu'à $n=0$)

$$\text{donc} \begin{cases} C_n = k + C_{n-1} \\ C_0 = k \end{cases} \longrightarrow C_n = kn$$

Bilan : · meilleur des cas : complexité constante
· pire des cas : complexité $k \times n$

↳ Conclusion : Complexité de la fonction

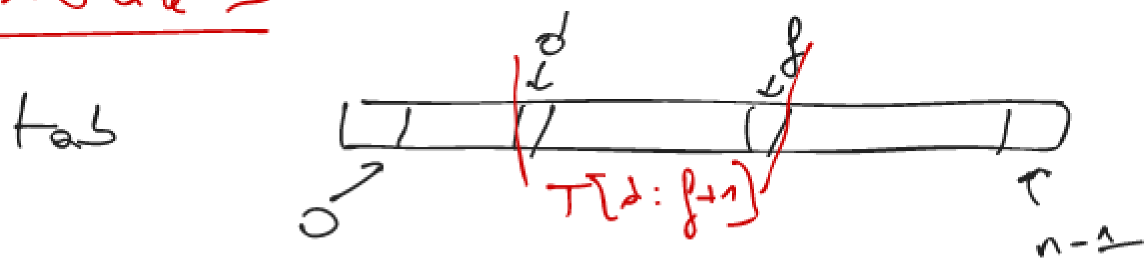
$$\begin{cases} \Omega(1) & : \text{meilleur des cas} \\ O(n) & : \text{pire cas} \end{cases}$$

Bilan : Complexité

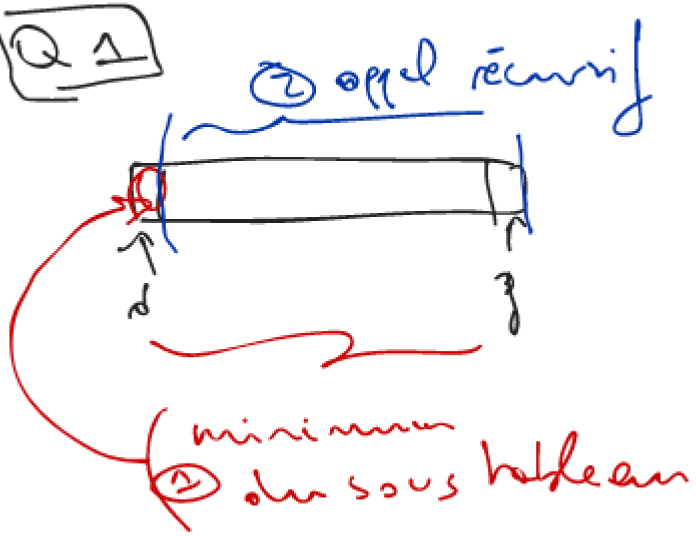
1. → Complexité du meilleur cas : Ω
   → Complexité du pire cas : O

2. Si pas le meilleur ou le pire cas : Θ

Exercice 3



Q 1



$T[d:f+1]$: sous-tableau entre les indices d et f inclus.

triSelectionRec(T, 0, 6)   T = [3, 1, 8, 5, 1, 4, 3]

Affichages :

Appel T[0,6] = [3, 1, 8, 5, 1, 4, 3]   (1)

Appel T[1,6] = [3, 8, 5, 1, 4, 3]   (2)

Appel T[2,6] = [8, 5, 3, 4, 3]   (3)

Appel T[3,6] = [5, 8, 4, 3]   (4)

Appel $[4,6] = [8,4,5]$ (5)

Appel $[5,6] = [8,5]$ (6)

appel récursif à triSelection$(T,6,6)$
sans affichage et qui termine
car $d=f$

En sortie $T[5,6] = [5,8]$ (6)

En sortie $T[4,6] = [4,5,8]$ (5)

En sortie $T[3,6] = [3,4,5,8]$ (4)

En sortie $T[2,6] = [3,3,4,5,8]$ (3)

En sortie $T[1,6] = [1,3,3,4,5,8]$ (2)

En sortie $T[0,6] = [1,1,3,3,4,5,8]$ (1)

→ le tableau est trié !

$\boxed{\text{Q2}}$

Posons $m = f - d$.

Soit $P(m)$ "Si tab est un tableau de taille n, et $0 \leq d \leq f \leq n-1$ t.q $f - d = m$, alors l'appel triSelectionRec(tab, d, f) termine, et trie le sous tableau tab[d f] en ordre croissant"

Montrons $P(m)$ par récurrence faible pour $m \geq 0$

→ Base Si $m = 0$, $d = f$, tab[d f] est trié (c'est un seul élément) Et la fonction ne fait rien quand $d = f$. Donc $P(0)$ est vérifié.

→ <u>Induction</u> : Soit $m \geq 1$. Supposons $P(m-1)$, Montrons que $P(m)$ est vérifié.

Soit tab un tableau de taille n, et d et f tels que $0 \leq d \leq f \leq n-1$ et $d - f = m$.

$m \geq 1$ donc $d < f$, donc on entre dans le if.

Le minimum de tab[d..f] est placé en tab[d]. Par hypothèse de récurrence, l'appel récursif trie tab[d+1...f] dans l'ordre croissant

(HR appliqué $f-(d+1) = m-1$, $P(m-1)$)

Donc la fonction triSelectionRec(tab, d, f) termine et trie tab entre d et f par ordre croissant

<u>Conclusion</u> : La propriété est vérifiée par récurrence faible.

TD4 suite

Exercice 3

Q3

- On pose $n = f - d$
- Soit $c(n)$ le nombre de comparaisons effectuées par la fonction triSelectionRec sur une entrée de taille n.

$\rightarrow c(0) = 0$

$\rightarrow c(n) = \underbrace{c_{rechercheMin}(n)}_{= n} + c(n-1)$

$$\begin{cases} c(n) = n + c(n-1) \\ c(0) = 0 \end{cases} \rightarrow \text{On résout par } \underline{\text{substitution}}$$

$$\begin{aligned} c(n) &= n + \underline{c(n-1)} \\ &= n + (n-1) + c\underline{(n-2)} \\ &= n + (n-1) + (n-2) + c(n-3) \\ &= \dots \end{aligned}$$

$$c(n) = \sum_{i=1}^{n} i + \cancel{c(0)}_{=0} = \frac{n(n+1)}{2}$$

Donc $\boxed{c(n) \text{ est en } \Theta(n^2)}$