Algo II : TD2

Exercice 1 :

* $T(n) = O(f(n)) \Leftrightarrow \exists c>0, \exists n_0>0, \forall n \geq n_0 :$
$T(n) \leq f(n)$

* $T(n) = \Theta(f(n)) \Leftrightarrow \exists c_1, c_2 > 0, \exists n_0 > 0$
$\forall n \geq n_0 : c_2 f(n) \leq T(n) \leq c_1 f(n)$

1) $P(n)$ est un poly de degré k

si $\exists a_0\ a_1 \dots a_m \in \mathbb{R}$ tq

$P(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_1 n + a_0$, $n \in \mathbb{N}$

$p(n) \leq \max_{0 \leq i \leq k}(a_i)\left(n^k + n^{k-1} + \dots + 1\right)$

$\leq \max_{0 \leq i \leq k}|a_i| \underbrace{\left(n^k + n^k + \dots + n^k\right)}_{k+1} \quad \forall n \geq 1$

$\leq \underbrace{\left((k+1)\max|a_i|\right)}_{c} n$

Par $n_0 = 1$ et $c = (k+1)\max_{0 \leq i \leq k}|a_i|$

on a : $p(n) \leq c\, n^k$; $\forall n \geq n_0$

$-|a| \leq a \leq |a| \Rightarrow n - |a| < n + a \leq n + |a|$

$n + |a| \leq c_1 n \Leftrightarrow (c_1 - 1) n \geq |a|$

$\Leftrightarrow n \geq \frac{1}{c_1 - 1}|a|$

$\exists n_0\ \forall n \geq n_0$

$0 < c_1 - 1 < 1 \Rightarrow \frac{1}{c_1 - 1} > 1 \Rightarrow \frac{|a|}{c_1 - 1} > |a|$

$1 < c_1 < 2$

pour $c_1 = \frac{3}{2}$ on a $n \geq \frac{|a|}{1/2} \geq 2|a|$

Pour $n_0 \geq 2|a|$ et $c = \frac{3}{2}$ on a :

$n + a < n + |a| \leq c_1 n$

* $n + a \geq n - |a|$

$n - |a| \geq c_2 n \Leftrightarrow (1 - c_2) n \geq |a|$

$\Leftrightarrow n \geq \frac{1}{1 - c_2}|a| \geq 1$

$0 < 1 - c_2 < 1 \Rightarrow \frac{1}{1 - c_2} > 1 \Rightarrow \frac{|a|}{1 - c_2} > |a|$

$0 < c_2 < 1$

$c_2 = \frac{1}{2}$ on a $n \geq 2|a|) n_0$

pour $n_0 \geq 2|a|$ et $c_2 = \frac{1}{2}$ on a :

$c_2 n \leq n - |a| < n + a$

Conclusion :

pour $n_0 = 2E|a|$; $c_1 = \frac{3}{2}$ et $c_2 = \frac{1}{2}$ on a :

$c_2 n < n + a \leq c_1 n$

$\Rightarrow \underbrace{c_2^b}_{c_2} n^b \leq (n+a)^b \leq \underbrace{c_1^b}_{c_1} n^b$; $\forall n \geq \underbrace{n_0}_{2E(|a|)}$

3°)

$n! = 1 \times 2 \times 3 \times \dots \times n \leqslant \underbrace{n \times n \times \dots \times n}_{n \text{ fois}}$

$n! \leqslant n^n \Rightarrow \log n! \leqslant n \log n, \forall n \geqslant 1$

$n! = \left(1 \cdot 2 \dots \frac{n}{2}\right)\left(\frac{n}{2}+1\right)\left(\frac{n}{2}+2\right) \dots \left(\frac{n}{2}+\frac{n}{2}\right)$

$n! \geqslant \left(\frac{n}{2}+1\right)\left(\frac{n}{2}+2\right) \dots \left(\frac{n}{2}+\frac{n}{2}\right)$

$\geqslant \underbrace{\frac{n}{2}\ \frac{n}{2} \dots \frac{n}{2}}_{\frac{n}{2}}$

$n! \geqslant \left(\frac{n}{2}\right)^{\frac{n}{2}}$

$\Rightarrow \log n! \geqslant \frac{n}{2} \log \frac{n}{2} \geqslant c\,(n \log n)\ \forall n \geqslant n_0$

$n \log \frac{n}{2} \geqslant \frac{1}{2} n \log n$

$\Leftrightarrow n \log n - n \log 2 - \frac{1}{2} n \log n \geqslant 0$

$\Leftrightarrow \frac{1}{2} n \log n - n \log 2 \geqslant 0$

$\Leftrightarrow n\left(\frac{1}{2}\log n - \log 2\right) > 0$

$\Rightarrow \frac{1}{2}\log n \geqslant \log 2$

$\Rightarrow n^{1/2} \geqslant 2$

$\Rightarrow n \geqslant 4$

et

$\log n! \geqslant \frac{1}{2}\left(\frac{1}{2} n \log n\right)$

$\geqslant \frac{1}{4} n \log n$

$c = \frac{1}{4}$ et $n_0 = 4$

Exercice 2

1°) l'algo A_1 est en $O(n^2)$

" " A_2 est de l'ordre de n (ou en $O(n)$)

or $n \leqslant n^2$ donc $O(n) \subseteq O(n^2)$

selon la formule l'algo A_2 est meilleur que A_1

2°) pour $n = 10$

$T_1(n) = 900$

$T_2(n) = 696$

$\Rightarrow A_1$ est meilleur que A_2 pour $n = 10$

3°) comparaison de T_1 et T_2

$9n^2 \overset{?}{\geqslant\leqslant} 100n + 96$

$\Rightarrow 9n^2 - 100n - 96 = 0$

$\Rightarrow n_0 = 12$

l'algorithme A_2 est plus efficace que A_1 à partir du rang $n = 12$

càd :

$\forall n \geqslant 12$ on a $100n + 96 \leqslant 9n^2$

4°)

la complexité de A_3 est la somme de la complexité de A_1 et de A_2

Soit $T_3(n)$ le coût de A_3

$T_3(n)$ vérifie :

$T_3(n) = T_1(n) + T_2(n)$

$= O(n^2) + O(n)$

$= O(\max(n, n^2))$

$= O(n^2)$ (pour n assez grand)

car $O(f+g) = O(f) + O(g) = O(\max(f, g))$

1°)

en C : on code les couleurs par

bleu → 0

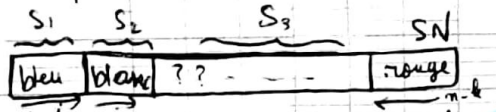
blanc → 2

rouge → 1

(si on code les bleus par 0, les blancs par 1 et les rouges par 2

→ on applique un Algo de tri → $O(n^2)$)

Supposon qu'à un moment donné on a :



$|S_1| = i$, $|S_2| = j$, $|S_4| = n-k$

bleu < blanc < rouge

(⚠ |s| indique la largeur de S)

tant que $S_3 \neq \emptyset$ faire :

prendre la 1ère de S_3

si c'est :

bleu : l'échanger avec la 1ère blanc de S_2 ($|S_1| = |S_1|+1$)

blanc : on fait rien ($|S_2| = |S_2|+1$)

rouge : l'échanger avec le dernier de S_3 ($|S_4| = |S_4|+1$)

ftq

on choisit k tq : $|S_1 S_2 S_3| = k$

$|S_4| = n-k$

$|S_1| + |S_4| + |S_2| + |S_4| = n$

$i + j + |S_3| + k' = n$

$|S_3| = (n-k') - (i+j)$

```
i:=0 ; j:=0 ; k:=n ;
tant que i+j+1 ≤ k faire
    cas où T[i+j+1] est :
        bleu : échanger (T[i+j+1], T[i+1]);
               i := i+1
        blanc : j := j+1
        rouge : échanger (T[i+j+1], T[k-1])
                k := k-1
ftantque
fin
```

La branche Tantque tourne n-1 fois (il y a n-1 itérations)

à chaque itération on fait :

4 opérations au maximum

3 → échanger 2 élém + 1 opération ($j \leftarrow i+1$; $k < k-1$)

$O(4) = 4\,O(1) = O(1)$

la complexité de l'algo est égale à

$n(O(1)) = O(n)$

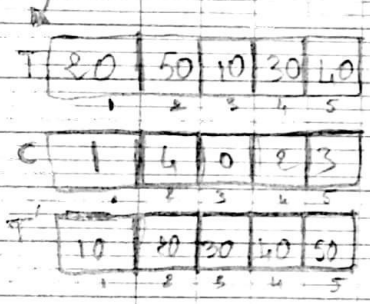
2) a -

```
comptage (T, n)
Début
    pour i := 1 à n faire
        //T[i] fixe
        cpt := 0;
        pour k := 1 à n faire
            si T[k] < T[i] alors
                cpt ← cpt + 1;
            fsi
```

fpour
C[i] ← cpt
fpour
retourner (C, n)
Fin

b-

Tri par comptage (T, n)
début
C := Comptage (T, n)
pour i := 1 à n faire
T'[C[i]+1] := T[i]
fpour;
Retourner (T', n)
Fin



c-

complexité du tri par comptage :

$$O(n^2) + O(n) = O(\max(n, n^2)) = O(n^2)$$

- $\sum_{i=1}^{n} \left(a(i) + \underbrace{\sum_{i=1}^{n} O(1)}_{O(n)} \cdot O(1) \right) = O(n^2)$ pour comptage
- $O(n^2)$
- $O(n)$

Exercice 4

* **pgcd itératif**

```
tantque a mod b ≠ 0 faire
    r ← a mod b;
    a ← b;
    b ← r;
ftq
retourner (b);
```

* **pgcd récursif:**

on suppose que $a \geq b$ et $(b > 0)$

```
pgcd R (a, b)
Début
    si a mod b = 0 alors
        retourner (b)
    sinon
        retourner (pgcd R (b, a mod b))
    fsi
fin
```

Méthode 2

pgcd(8,6) = pgcd(2,6)
= pgcd(2,4)
= pgcd(2,2)

```
pgcd (a, b)
Début
    si a=b alors retourner (a)
    sinon
        si a>b alors retourner (pgcd(a-b, b))
        sinon retourner (pgcd(a, b-a))
        fsi
fin  fsi
```

b-

```
Sauith (n)
Début
    si n=0 alors retourner (
    sinon Retourner (Sauith(n-1)+n)
    fsi
Fin
```

⚠

$S_n = \underbrace{1+2+\dots+n-1}_{S_{n-1}} + n$

$$S_n = \begin{cases} 0 & \text{si } n=0 \\ S_{n-1} + n & \text{si } n>0 \end{cases}$$

```
Somme chiffre (n)
Début
    si n=0 alors retourner (0)
    sinon
        retourner (n mod 10 + somme chiffre (n div 10))
    fsi
fin
```

SC(123) = 3 + SC(12)
2 + SC(1)
1 + SC(0)
0

Somme des chiffres pairs de n

```
Som_chif_pair (n)
Début
    si n=0 alors retourne (0)
    sinon
    Si (n%10) %2 =0 alors
    Retourner (n%10 + Somm_chif_pair(n div 10));
    Sinon
        Retourner (som_chif_pair (n div 10))
    fsi
fsi
fin
```

C-

```
NB_zero (T,n)
début
    si n=0 alors retourner (0);
    Sinon
        si T[n]=0 alors
            retourner (1+NB_zero(T,n-1))
        Sinon
            retourner (NB_zero (T,n-1))
        fsi
    fsi
Fin
```

Solution 2 :

```
NB-Z(T,i,n)
Début
    si i≤n alors
        si T[i]=0 alors
            retourner (1+NB_Z(T,i+1,n))
        Sinon
            retourner (NB_Z(T,i+1,n))
        fsi
    Sinon
fin
```

Exercice 7 :

I = $O(1)$, II = $O(n^2)$, III = $O(n)$, IV = $O(n \log_2 n)$

V = $O(2^n)$

classement : I , IV , III , II , V

d) on a $n^{0,999} \log n < n^{0,999} \cdot n^{0,001} = n$ (on a $\forall x \geq 1, \forall a \in]0;1[$, $\log x \ll x^a$)

$$E_n = \frac{x_1 + x_2 + \dots + x_{n-1}}{n} + \frac{x_n}{n}$$

$$= \frac{1}{n}(x_1 + x_2 + \dots + x_{n-1}) + \frac{x_n}{n}$$

$$= \frac{n-1}{n} E_{n-1} + \frac{x_n}{n}$$

avec

$$E_{n-1} = \frac{x_1 + \dots + x_{n-1}}{n-1} \Rightarrow (x_1 + \dots + x_{n-1}) = (n-1) E_{n-1}$$

moyenne (X , n)
Début
  si n = 1 alors
    retourner (X[1])
  sinon
    Retourner (X[n]/n + ((n-1)/n) * moyenne (X , n-1))
  fin si
fin

Exercice 6 :

1) $2^{n+1} = O(2^n)$

On a $\lim_{n \to +\infty} \frac{2^{n+1}}{2^n} = 2$ donc $2^{n+1} = O(2^n)$ est vraie

2) $2^{2n} = O(2^n)$ faux

par absurde : Supposons qu'il est vraie

donc $\exists c > 0, \exists n_0 > 0, \forall n > n_0$ tq :

$$2^{2n} \leq c\, 2^n \Rightarrow c \geq 2^n$$

contradiction : c doit être indépendante de n

3) $2^{2n} = O(n!)$

soit $n \in \mathbb{N}$

soit $w_n = \frac{2^n}{n!}$ $\forall n$

on a $\frac{w_{n+1}}{w_n} = \frac{2^{n+1}}{(n+1)!} \times \frac{n!}{2^n} = \frac{4}{n+1} < 1$

alors la suite w_n converge à partir d'un certain rang . On a donc : $2^{2n} = O(n!)$