Algo II : TD 1 :

Exercice 1 :

⚠ Schéma d'un algo :

<nom de la fct> <paramètres> Données : (...)
Début
⋮
← retrouver (résultat)
Fin

Résultats : ....
Début
⋮
Fin
} Pas de retrouver

Algorithme plus grand entier P

Données : n : entier strictement positif
Résultat : p : entier
Variables : i : entier

Début
p ← 0
i ← 1
tant que i ⩽ n
p ← p+1
i ← 10 × i
Fin tant que
p ← p-1 → le nbre d'itération est p+1
Fin

Ou :

plus grand entier p(n)
Début
i = 1 ; p = 0
tant que i ⩽ n faire
p := p+1
i := 10 * i
ftant que (nbre d'itération est p)

Soit p la dernière itération, On a :
$10^p > n$ et $10^{p-1} \leqslant n \Rightarrow 10^{p-1} \leqslant n < 10^p$

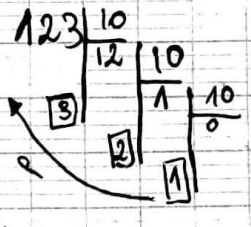
$\Rightarrow p-1 \frac{\ln 10}{\ln 10} \leqslant \frac{\ln(n)}{\ln(10)} \leqslant p \frac{\ln 10}{\ln 10}$

$\Rightarrow p-1 \leqslant \log_{10}(n) \leqslant p$

$\Rightarrow p-1 = E(\log_{10} n)$
$\Rightarrow p = E(\log_{10} n) + 1$
= le nbre de chiffres de n

Alors le nbre de bits nécessaire pour représenter un entier n est $E(\log_{10} n) + 1$

• Autre Méthode :



Début
P = 0
tant que n ⩾ 0 faire
→ écrire (n mod 10)
n := n div 10
p := p+1
ftant que
← retourner (p)

SMIA STUDIES

Exercice2

Soient n_1 et n_2 deux entiers positifs ayant n chiffres chacun.
Chaque entier n_i est représenté par un tableau de n chiffres

$T_i[1...n]$
(on place le chiffre des unités à la position n dizaine — $n-1$)

Algorithme pour calculer la somme chiffre à chiffre, de deux tableaux

$T_1[1...n]$ et $T_2[1...n]$
Somme Tableau chiffre (T_1, T_2, n)
Début

```
r := 0;
Pour i := n à 1 (pas1) faire
    T3[n-i+1] ← (T1[i] + T2[i] + r) mod 10
        (pour obtenir unité)
    r ← (T1[i] + T2[i] + r) div 10
fpour
Si r ≠ 0 alors n := n+1
              T3[n] := r;
fsi
```

// tableau T_3 inversé

```
Pour i = 1 à n/2 faire
    X := T3[i];
    T3[i] := T3[n-i+1];
    T3[n-i+1] = X
fpour
retourner (T3, n)
fin
```

Exercice 3:

1°/ Intersection (T_1, T_2, n_1, n_2)
1ère Méthode:

```
Début
    k ← 1
    pour i := 1 à n1 faire
        chercher si T1[i] est dans T2
        j ← 1
        tantque (j ≤ n2) et (T1[i] ≠ T2[j]) faire
            j ← j+1
        fintantque
        si (j ≤ n2) alors T3[k] := T1[i];
                          k := k+1;
        fsi
    fimpour
retourner (T3, k-1)
Fin
```

2ème Méthode

```
// fct qui cherche un élé x dans
   un tableau T à n éléments
Localiser (x, T, n) (: booléen)
Début j ← 1
```

```
tant que (j<n) et (x≠T[j]) faire
        j ← j+1
ftantque

si (j≤n) alors retourner (vrai)
   sinon retourner (faux)
fsi
Fin
```

```
Intersection (T1, T2, n1, n2)
  Début
      k ← 0
      Pour i:=1 à n1 faire
          si localiser(T1[i], T2, n2) alors
              k ← k+1 ; T3[k] := T1[i]
          fsi
      fpour
  retourner (T3, k)
  Fin
```

2°/

E ens fini

$\forall x \in E, \forall A \subseteq E$

$$\chi_A(x) = \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si } x \notin A \end{cases}$$

$E = \{1,2,3,4,5\}$

$A = \{2,4\} \Leftrightarrow T_1$

$B = \{1,4,5\}$ T_2

	1	2	3	4	5
T_1	F	V	F	V	F
T_2	V	F	F	V	V
	F	F	F	V	F

```
Intersection Ens (T1, T2, n)
  Début
      Pour i:=1 à n faire
          si (T1[i] et T2[i])
              alors T3[i] := Vrai
              sinon T3[i] := faux
          fsi
      fpour
      retourner (T3, n)
  Fin
```

Cette Algo est en $O(n)$

Exercice 4 :

Algo pour construire un carré magique

```
carrée magique (n)
Début
  // initialisation de la matrice
  pour i:=1 à n faire
      pour j=1 à n faire
          A[i,j] := 0
      fpour
  fpour
  // construction
  i := (n+1)/2 - 1 ; j = (n+1)/2
  A[i,j] := 1 ;
  Pour k := 2 à n² faire
      i := i-1 , j := j+1
      si i=0 alors i=n ; fsi
      si j := n+1 alors j=1 fsi
      tantque A[i,j] ≠ 0 faire
          i := i-1 ;
          j := j-1 ;
          si i = 0 alors i = n fsi
          si j = 0 alors j := n fsi
      ftantque
```

A[i,j] = k

fin pour

retourner [A,n]

Fin

Exercice 5

Preuve de l'invariant par récurrence sur le nbre n :

pour n = 0 on a :

$n \geq 0$ et $m_0 = 2^0 b = b$ et $b < 2a$

(l'invariance est vrai à l'initialisation)

H.R (on suppose que l'initialisation est vrai pour n = k)

$k \geq 0$ et $m_k = 2^k b$ et $m_k \leq 2a$

et $m_k \leq a$

Pour n = k+1 on a

$m_{k+1} = 2 m_k \Rightarrow m_{k+1} = 2 \cdot 2^k b = 2^{k+1} b$

$m_{k+1} = 2 m_k$ et $m_k < a \Rightarrow m_{k+1} \leq 2a$

Clc : l'invariant est vrai pour toute itération n

Exercice 6 :

i	s	j
1	0	
	1	1
2	0	1
	2	
	4	2
3	3	1
	5	2
	7	
	9	3

4	8 10 12 14 16	1 2 3 4

l'algo calculer n^2

La boucle contrôlée par j est en O(i)

2°/ $O(n^2)$ car : La boucle contrôlée par i en $O(n^2)$ car

$\sum_{i=1}^{n} O(i) = O\left(\sum_{i=1}^{n} i\right) = O\left(\frac{n(n+1)}{2}\right) = O(n^2)$

Exercice

A1) à l'itération k on a $i = 2^k$

Soit p la dernière itération de tant que on a : $i = 2^p$

et $2^p > n$ et $2^{p-1} \leq n$

On a $2^{p-1} \leq n < 2^p$

$\Rightarrow p-1 \leq \log_2 n < p$

$\Rightarrow p - 1 = E(\log_2 n)$

$\Rightarrow p = E(\log_2 n) + 1$

instruction	
i := 1	O(1)
s := 0	O(1)
tant que i ≤ n faire	
s := s + 1 ;	$O(\log_2 n)$
i := 2*i ;	
fin tant que	

On a $T(n) = T(\text{Nbre-itération} \times (T(i \leq n) + T(A,i)))$

de tant que

$= O((E(\log_2 n) + 1) \times O(1))$

$= O(E(\log_2 n) + 1)$

T(n) s'écrit comme : $a(E(\log_2 n) + 1) + n^d$

avec : a = 1

b = 2 (car c'est log2)

d = 0

comme $a = b^d$

donc $T(n) = O(\log_2 n)$