Excercice 37 :

A se réduit à B si $\exists f$ totale, calculable telle que $\forall x \in \mathbb{N}$, $x \in A$ ssi $f(x) \in B$.

B décidable et $A \propto B \Rightarrow A$ décidable

$\Rightarrow$

```
int fca(int x) {
        return fcb(f(x));
}
```

car B est décidable.

B récursivement énumérable et $A \propto B \Rightarrow A$

récursivement énumérable

$\exists f_B \qquad B = \{ f_B(0), f_B(1), \dots, f_B(i), \dots \}$

f_B totale et calculable.

$x \in A$ ssi $f_A(x) \in B$

$\Rightarrow \exists i$ tel que $f_A(x) = f_B(i)$

```
int fA ( ) {
    Pour tout couple (x, i) {
        Si fA(x) = fB(i) {
            afficher fB(i);
        }
    }
}
```

Exercice 32 :

Application $f : A \to B$ $\forall x \in A : P_f(x)$

$= \lim\limits_{n \to +\infty} f^n(x)$ si la limite existe.

indéfini sinon.

$f : \mathbb{N} \to \mathbb{N}$ $\quad x \to x - 1$ $\quad \forall x \ P_f(x) = 0$

$f : \mathbb{N} \to \mathbb{N}$ $\quad x \to x - 1$ si x est impair

$\quad x \to x$ si x est pair

$P_f(x) = 2\left\lfloor \frac{x}{2} \right\rfloor = f(x)$

$x, f(x), f^2(x), f^3(x) \ldots, f^n(x)$

$\exists n : f^n(x) = f^{n+1}(x) \Rightarrow Pf(x) = f^n(x)$

1. $x \to x+1$ $\quad \forall x : Pf(x) =$ indéfini

2. $x \to 0 \times x$ $\quad \forall x : Pf(x) = 0$

3. $x \to \begin{cases} x/2 & \text{si } x \text{ pair} \\ x & \text{sinon} \end{cases}$ $\quad \forall x :$ x s'écrit de façon unique sous la forme $2^i y$ avec y impair $Pf(x) = y$

4. $x \to \begin{cases} x & \text{si } x \geq \overset{\text{cst}}{n} \\ x+1 & \text{sinon} \end{cases}$ $\quad \forall x : Pf(x) = \max(x, n)$

5. $x \to \begin{cases} x & \text{si } x \geq n \\ x-1 & \text{sinon} \end{cases}$ $\quad Pf(x) = x$ pour $x \geq n$

6. $(x,y) \to \begin{cases} (x-1, y+1) & \text{si } x > 0 \\ (x,y) \end{cases}$

$Pf((x,y)) = 0, (x+y)$

D

Excercice 33 :

4. $f_0 = 0$, $f_1 = 1$, $\forall n > 1$

$$f_n = f_{n-1} + f_{n-2}$$

```
int fib(int n){
    if(n<2) return n;
    else return fib(n-1) + fib(n-2);
```

Montrer que le nombre d'addition est compris entre $\sqrt{2}^n$ et 2^n

t_n = nb additions pour calculer fib(n)

$t_0 = t_1 = 0$

$n \geq 2 \qquad t_n = 1 + t_{n-1} + t_{n-2}$

$t_n \leq v_n \qquad v_n = 1 + 2 v_{n-1} \quad O(2^n)$

$u_n \leq t_n \qquad u_n = 1 + 2 u_{n-2} \quad O(\sqrt{2}^n)$

$u'_0 = 1 \quad , \quad u'_n = 2 u'_{n-2}$

Supposons n pair

$u'_n = 2 \cancel{u'_{n-2}}$

$\cancel{u'_{n-2}} = 2 \cancel{u'_{n-4}}$

$\cancel{u'_{n-4}} = 2 \cancel{u'_{n-6}}$

$\dots$

$\cancel{u'_2} = 2 u'_0$

$u'_n = 2^{\frac{n}{2}} = \left(2^{\frac{1}{2}}\right)^n = \sqrt{2}^n$

$$v_n = 1 + 2 v_{\cancel{n}-1}$$

$$v_{\cancel{n}-1} = 1 + 2 v_{\cancel{n}-2}$$

$$\cancel{} \quad - - - - \quad \cancel{}$$

$$\cancel{v_1} = 1 + 2 v_0$$

$$v_n = n + 2^n \times \varepsilon$$

$$w_n = 1 + 2 w_{\cancel{n}-2}$$

$$w_{n-1} = 1 + 2 w_{\cancel{n}-4}$$

$$\cancel{} \; . - - - - .$$

$$w_{\cancel{n}-4} = 1 + 2 w_{n-10}$$

$$- - - -$$

$$w \; .... \quad n + \left| \sqrt{2}^{\,n} \times \varepsilon \right|$$

donc $t_n \approx \left(\frac{1+\sqrt{5}}{2}\right)^n$

nb d'or puissance n.

5. Fib [0] = 0 ; Fib [1] = 1;

```
for (int i = 2 ; i ≤ n ; i++) {
    F [i] = F [i-1] + F [i-2];
return F [n];
```

$O(n)$

Programmation dynamique.

la taille de la donnée est $O(\log(n))$ donc cet algo n'est pas polynomial en la taille de la donnée.

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix}$$

$n = 1$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} f_2 & f_1 \\ f_1 & f_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

$n-1 \Rightarrow n$:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} f_n & f_{n-1} \\ f_{n-1} & f_{n-2} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} f_n & f_{n-1} \\ f_{n-1} & f_{n-2} \end{pmatrix} \begin{pmatrix} f_n + f_{n-1} & f_n \\ f_{n-1} + f_{n-2} & f_{n-1} \end{pmatrix}$$

$$= \begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix}$$

Pour calculer f_n, il suffit de calculer :

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

$A \times B : O(1)$

si A et B sont des matrices 2×2, il faut 8 mult et 4 additions.

$$A^9 = A^8 \times A = \left(\left(A^2\right)^2\right)^2 \times A$$

$$A^n = \left(A^{\lfloor \frac{n}{2} \rfloor}\right)^2 \times A \quad \text{si } n \text{ impair}$$

Puissance (x, n) :

aux = Puissance $(x, n/2)$

Si n est pair return aux * aux.

Sinon return aux * aux * x

Le résultat est exponentiel en la taille de la donnée, donc il ne peut exister d'algo polynomial en la taille de la donnée.

Excercice 34 :

$G = (V, E)$

V : vertex — sommet

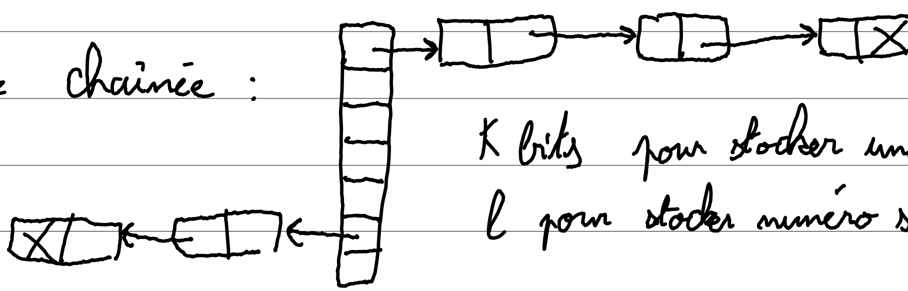
E : edge — arête

$|V| = n$

$|E| = m$

Matrice adjacence : $A[i,j] = \text{True}$ ssi $\{i, j\} \in E$

Liste chaînée :



K bits pour stocker une adresse

l pour stocker numéro sommets

	Taille	$e \in E$	Parcours
Matrice adjacence	$O(n^2)$	$O(1)$	$O(n^2)$
Liste chaînée	$O((m+n)K + ml)$	$O(\deg(G))$	$O(m)$

$\Theta(n) \leqslant m \leqslant O(n^2)$

$l = \log(m)$

$\log n \leqslant \log(m) \leqslant 2\log(n)$

car log est fonction croissante.

Exercice 35

$n = ab$ et n premier.

$\min(a,b) \leqslant \sqrt{n}$

Si $\min(a,b) > \sqrt{n})$ alors $ab > n$

Il existe un algo en $O(\sqrt{n})$, cet algo n'est pas polynomial.

n est-il le produit de 2 nombres premiers ?

$\exists a, b$ premiers tq $n = ab$.

il existe un algo en $\sqrt{n}$.

Cet algo n'est pas polynomial car $\sqrt{n}$ est exponentiel en $\log(n)$.

Existe t'il un algo polynomial ?

On ne sait pas.

Exercice 36 :

tous les 4 ans, la puissance des ordis est multipliée par 8.

A : $O(n^3)$ $\quad T_A(10) = 1s$ il y a 40 ans

B : $O(2^n$ $\quad T_B(10) = 1s$ il y a 40 ans

$T_A(n) = \alpha\, n^3$, $T_B(n) = \beta\, 2^n$

$$\alpha\, 10^3 = \alpha\, 10000$$

$$\alpha = \frac{1}{10000}$$

$$T_A(n)_{auj} = \frac{1}{10000} \times \frac{1}{8^{10}} \times n^3$$

on peut auj, traiter en 1s un pb de taille 10000.

$$T_B(10) = \beta \times 2^{10} = 1$$

$$\beta = \frac{1}{2^{10}}$$

$$T_B(n) = \beta \times \frac{1}{8^{10}} \times 2^n$$

$$= \frac{1}{2^{10}} \times \frac{1}{8^{10}} \times 2^n$$

On peut auj, traiter en 1 s un pb de taille 40.

P : on peut dire qu'il existe ou non un algo polynomial.

il existe algo Non déterministe polynomial.

NP : On peut montrer que la réponse est OUI.