I Partie théorique

I.1

$$\begin{aligned}\begin{cases}P &= a_0 + a_1X + \cdots + a_nX^n\\ Q &= b_0 + b_1X + \cdots + b_nX^n\end{cases}\\ P + Q = (a_0 + b_0) + \cdots + (a_n + b_n)X^n\end{aligned}$$

Pour $k \in [\![0, n]\!]$, on calcule $a_k + b_k$ On a ainsi $\Theta(n)$ additions. $\boxed{\text{C'est optimal}}$ car il y a $n + 1$ coefficients.

I.2

$$\begin{aligned}\begin{cases}P &= a_0 + a_1X + \cdots + a_nX^n\\ Q &= b_0 + b_1X + \cdots + b_nX^n\end{cases}\\ P \times Q = c_0 + c_1X + \cdots + c_{2n}X^{2n}\end{aligned}$$

où

$$c_n = \sum_{k=0}^{n} a_k b_{n-k}$$

Pour $k \in [\![0, 2n]\!]$ on calcule c_k (à l'aide d'une boucle `for` par exemple)

Complexité

$$\begin{aligned}\sum_{k=0}^{2n} \underbrace{\Theta(k)}_{k+1 \text{ produits, } k \text{ sommes}} &= \Theta((2n)^2)\\ &= \Theta(n^2)\end{aligned}$$

On va améliorer : on utilise la stratégie romaine, diviser pour régner .

I.3

$$\begin{aligned}P &= a_0 + a_1X + \cdots + a_{\lceil \frac{n}{2} \rceil - 1}X^{\lceil \frac{n}{2} \rceil - 1} + a_{\lceil \frac{n}{2} \rceil}X^{\lceil \frac{n}{2} \rceil} + \cdots + a_nX^n\\ Q &= b_0 + b_1X + \cdots + b_{\lceil \frac{n}{2} \rceil - 1}X^{\lceil \frac{n}{2} \rceil - 1} + b_{\lceil \frac{n}{2} \rceil}X^{\lceil \frac{n}{2} \rceil} + \cdots + b_nX^n\\ P &= AX^{\lceil \frac{n}{2} \rceil} + B\\ Q &= CX^{\lceil \frac{n}{2} \rceil} + D\\ PQ &= ACX^{2\lceil \frac{n}{2} \rceil} + (AD + BC)X^{\lceil \frac{n}{2} \rceil} + BD\end{aligned}$$

Cas d'arrêt $\boxed{n = 0}$

Relation de récurrence

$$c_n = \underbrace{4c_{\lceil \frac{n}{2} \rceil}}_{\text{appels récursifs}} + \underbrace{\Theta(n)}_{\text{sommes}} + \underbrace{\Theta(n)}_{\text{décalages (multiplication par } X^{\cdots})}$$

Complexité

$$c_n = ac_{\lceil \frac{n}{2} \rceil} + \Theta(n^\beta)$$
$$\alpha = \lg a \Leftrightarrow a = 2^\alpha$$

Ici, $\alpha = \lg 4 = 2\lg 2 = 2 > \beta = 1$

$$c_n = \Theta(n^\alpha) = \Theta(n^2)$$

Bi~~h~~lan On retrouve la complexité quadratique de la méthode naïve :/

I.4

$$(AD + BC) = (A + B)(C + D) - AC - BD$$

$$P \times Q = \underbrace{AC}_{\text{1er produit}} X^{2\lceil \frac{n}{2} \rceil} + \left(\underbrace{(A+B)(C+D)}_{\text{2e produit}} - \underbrace{AC}_{\text{1er produit}} - \underbrace{BD}_{\text{3e produit}} \right) X^{\lceil \frac{n}{2} \rceil} + \underbrace{BD}_{\text{3e produit}}$$

Relation de récurrence

$$c_n = \underbrace{3c_{\lceil \frac{n}{2} \rceil}}_{\text{appels récursifs}} + \Theta(n^1)$$

Complexité

$$c_n = \Theta(n^\alpha) = \Theta(n^{\lg 3})$$

I.5

$$p = \overline{a_n a_{n-1} \dots a_0}^2 = \overline{a_n \dots a_{\lceil \frac{n}{2} \rceil} 0 \dots 0}^2 + \overline{0 \dots 0 a_{\lceil \frac{n}{2} \rceil - 1} \dots a_0}^2$$
$$q = \overline{b_n b_{n-1} \dots b_0}^2 = \overline{b_n \dots b_{\lceil \frac{n}{2} \rceil} 0 \dots 0}^2 + \overline{0 \dots 0 b_{\lceil \frac{n}{2} \rceil - 1} \dots b_0}^2$$

$$B = \overline{a_{\lceil \frac{n}{2} \rceil - 1} \dots a_0}^2 \qquad A = \overline{a_n \dots a_{\lceil \frac{n}{2} \rceil}}^2$$
$$D = \overline{b_{\lceil \frac{n}{2} \rceil - 1} \dots b_0}^2 \qquad C = \overline{b_n \dots b_{\lceil \frac{n}{2} \rceil}}^2$$

```
open Format

let deg = Array.length
type polynome = int array;;

let op_on_first_smaller _P _Q op =
        let result = Array.make (deg _Q) 0 in
        for k = 0 to (deg _P - 1) do
                result.(k) <- op _P.(k) _Q.(k)
        done;
        for k = (deg _P) to (deg _Q - 1) do
                result.(k) <- _Q.(k)
        done;
        result;;

let (+) _P _Q = if (deg _P) > (deg _Q)
        then op_on_first_smaller _Q _P (+)
        else op_on_first_smaller _P _Q (+);;

let (-) _P _Q = if (deg _P) > (deg _Q)
        then Array.map (fun x -> -x) (op_on_first_smaller _Q _P (-))
        else op_on_first_smaller _P _Q (-);;

assert (([| 3; 4; 5 |] + [| 3; 8; 10 |]) = [| 6; 12; 15 |]);
assert (([| 3; 4; 5 |] - [| 3; 8; 10 |]) = [| 0; -4; -5 |]);

let ( * ) _P _Q =
        let result = Array.make (deg _P + deg _Q - 1) 0 in
        for k = 0 to (deg _P + deg _Q - 1) do
                for i = 0 to k do
                        if i < (deg _P) && k-i < (deg _Q) then
                                result.(k) <- result.(k) + _P.(i) * _Q.(k-i)
                done;
        done;
        result;;

Array.map (printf "%d, ") ([| 1; 1; 1; 1; 1; 1; 1; 1 |] * [| -1; 1 |]);
```