TP 5 : FFT pour le produit de polynômes

1 Opérations dans un corps fini

Question 2. Les trois opérateurs.

```
let ( ++ ) x y=(x+y) mod p ;;
let ( ** ) x y=(x*y) mod p ;;
let ( -- ) x y=x ++ (p-y) ;;
```

Question 3. On a donc :

$$\mathrm{PGCD}(m,r) = um + vr = um + v(n - mq) = vn + (u - vq)m$$

D'où $(v, u - vq)$ est un couple de Bézout associé à n et m.

Question 4.

```
let rec bezout n m=match m with
  | 0 -> (1,0)
  | _ -> let q,r=n/m, n mod m in let u, v=bezout m r in v,u-q*v
;;
```

Question 5.

```
let ( // ) x y=
  let m,_=bezout y p in
  if m>0 then
    x ** m
  else
    x ** ((m mod p)+p)
;;
```

2 Opérations sur les polynômes et produit naïf

Question 6. On autorise un appel récursif pour supposer que `a` est de longueur supérieure à `b`.

```
let rec add_p a b=
  let n,m=Array.length a, Array.length b in
  if n<m then add_p b a else
    let c=Array.copy a in
    for i=0 to m-1 do
      c.(i) <- c.(i) ++ b.(i)
    done ;
    c
;;
```

Question 7.

```
let prod_mon a n=
  Array.append (Array.make n 0) a
;;
```

Question 8.

```
let prod_scal a x=
  Array.map (function y -> y ** x) a
;;
```

Question 9.

```
let prod_p a b=
  let c=ref [| |] in
  for i=0 to Array.length b -1 do
    c:= add_p !c (prod_mon (prod_scal a b.(i)) i)
  done ;
  !c
;;
```

3 Principe de la FFT et produit des évaluations

Question 10.

```
let prod_terme v w=
  let n=Array.length v in
  let z=Array.make n 0 in
  for i=0 to n-1 do
    z.(i) <- v.(i) ** w.(i)
  done ;
  z
;;
```

4 Racines de l'unité et évaluation

4.1 Racines de l'unité dans $\mathbb{F}_p$

Question 11.

```
let racines_unite=
  let v=Array.make 13 41 in
  for i=11 downto 0 do
    v.(i) <- v.(i+1) ** v.(i+1)
  done ; v;;
```

4.2 Divisons, pour régner ensuite

Question 12. *« Division ».* ***a.*** Le reste dans la division euclidienne de X^j par X^k-1 est X^j lui-même si $j<k$ et X^{j-k} si $j>k$, car $j-k<n-k=k$.

b.
- On vérifie facilement que $F-\sum_{j=0}^{k-1}(f_j+f_{j+k})X^j$ est divisible par X^k-1, et le polynôme $\sum_{j=0}^{k-1}(f_j+f_{j+k})X^j$ est de degré strictement inférieur à k, c'est donc le reste dans la division euclidienne de F par X^k-1. D'où $R_0(X)=\sum_{j=0}^{k-1}(f_j+f_{j+k})X^j$.
- De même, le reste dans la division euclidienne de X^j par X^k+1 est X^j si $j<k$ et $-X^{k-j}$ sinon. D'où $R_1=\sum_{j=0}^{k-1}(f_j-f_{j+k})X^j$.

c. On a $\overline{R_1}(X)=\sum_{j=0}^{k-1}(f_j-f_{j+k})w^jX^j$.

d.

```
let calculR0 f k=
  let v=Array.make k 0 in
  for i=0 to k-1 do
    v.(i) <- f.(i) ++ f.(i+k)
  done ;
  v
;;

let calculR1b f k w=
  let v=Array.make k 0 and q=ref 1 in
  for i=0 to k-1 do
    v.(i) <- (f.(i) -- f.(i+k)) ** !q ;
    q:= !q ** w
  done ;
  v
;;
```

Question 13. *« Règne ».* ***a.***

$$\begin{aligned} F(\omega^{2j}) &= \sum_{i=0}^{k-1} f_i\omega^{2ij} + \sum_{i=k}^{n-1} f_i\omega^{2ij} \\ &= \sum_{i=0}^{k-1} f_i\omega^{2ij} + \sum_{i=k}^{n-1} f_i\omega^{2(i-k)j} && \text{car } \omega^{2k} = \omega^n = 1 \\ &= \sum_{i=0}^{k-1} (f_i + f_{i+k})\omega^{2ij} && \text{(changement d'indice } i \leftarrow i+k \text{ à droite)} \\ F(\omega^{2j}) &= R_0(\omega^{2j}) \end{aligned}$$

De même,

$$\begin{aligned} F(\omega^{2j+1}) &= \sum_{i=0}^{k-1} f_i\omega^{2ij}\omega^i + \sum_{i=k}^{n-1} f_i\omega^{2ij}\omega^i \\ &= \sum_{i=0}^{k-1} f_i\omega^{2ij} + \sum_{i=k}^{n-1} f_i\omega^{2(i-k)j}(-\omega^{i-k}) && \text{car } \omega^k = -1 \\ &= \sum_{i=0}^{k-1} (f_i - f_{i+k})\omega^{2ij}\omega^i \\ F(\omega^{2j+1}) &= \overline{R_1}(\omega^{2j}) \end{aligned}$$

b. Il suffit « d'entrelacer » les éléments des deux k-uplets $\varphi_{\omega^2}(R_0)$ et $\varphi_{\omega^2}(\overline{R_1})$ pour obtenir $\varphi_\omega(F)$.

c.

```
let recomposition t0 t1 k=
  let n=2*k in
  let v=Array.make n 0 in
  for i=0 to k-1 do
    v.(2*i) <- t0.(i) ;
    v.(2*i+1) <- t1.(i)
  done ;
  v
;;
```

4.3 King in the north !

Question 14. *Un algorithme « Diviser pour Régner » pour la calcul de $\varphi_\omega(F)$.*

a. La représentation de $\varphi_1(F)$ est un tableau identique à la représentation de F (un seul élément !) dans ce cas.

b.

```
let rec evaluation f n w=
  if n=1 then f else begin
    let k=n/2 in
    let r0, r1b=calculR0 f k, calculR1b f k w in
    let w2 = w ** w in
    let t0, t1=evaluation r0 k w2, evaluation r1b k w2 in
    recomposition t0 t1 k
  end
;;
```

c. C'est essentiellement du cours, la relation est la même que pour le tri fusion.

5 Interpolation, et conclusion

Question 15.

```
let interpolation t n w=
  prod_scal (evaluation t n (1 // w)) (1 // n)
;;
```

Question 16. *Conclusion.*

```
let produitFFT a b n w=
  let pa, pb=evaluation a n w, evaluation b n w in
  interpolation (prod_terme pa pb) n w
;;
```