Computational Algebra. Lecture 8

M.Eulàlia Montoro

6 de abril de 2022

FFT,Cooley and Tukey (1965)

Recall $f = q_0 \cdot (x^{\frac{n}{2}} - 1) + r_0 = q_1 \cdot (x^{\frac{n}{2}} + 1) + r_1$. If $f = F_1 x^{\frac{n}{2}} + F_0$, then $r_0 = F_0 + F_1$ and $r_1 = F_0 - F_1$. Moreover

$$f(\omega^{2l}) = r_0(\omega^{2l})$$

and

$$f(\omega^{2l+1}) = r_1(\omega^{2l+1}) = r_1^*(\omega^{2l}), \quad r_1^* = r_1(\omega x)$$

Then

$$DFT_\omega(f) = (f(1), \ldots, f(\omega^{n-1})) =$$

$$= (r_0(1), r_1^*(1), \ldots, r_0(\omega^{n-2}), r_1^*(\omega^{n-2}))$$

FFT,Cooley and Tukey (1965)

Let us consider $\mathbb{F} = \mathbb{C}$, and ω a primitive n-root of the unity. Let $f = \sum_{i=0}^{n-1} f_i x^i \in \mathbb{C}[x]$ a polynomial of degree$< n = 2^k$. Then

$$DFT_\omega(f) = (f(1), \ldots, f(\omega^{n-1}))$$

Notice that

$$f(\omega^p) = \sum_{j=0}^{\frac{n}{2}-1} f_{2j}\omega^{2jp} + \omega^p \sum_{j=0}^{\frac{n}{2}-1} f_{2j+1}\omega^{2jp}$$

and $f(\omega^{p+\frac{n}{2}}) = \sum_{j=0}^{\frac{n}{2}-1} f_{2j}\omega^{2j(p+\frac{n}{2})} + \sum_{j=0}^{\frac{n}{2}-1} f_{2j+1}\omega^{(2j+1)(p+\frac{n}{2})} =$

$$= \sum_{j=0}^{\frac{n}{2}-1} f_{2j}\omega^{2jp} - \omega^p \sum_{j=0}^{\frac{n}{2}-1} f_{2j+1}\omega^{2jp}$$

and take $\alpha = \omega^2$.

FFT,Cooley and Tukey (1965)

If $R_0 = \sum_{j=0}^{\frac{n}{2}-1} f_{2j} x^{2j}$ and $R_1 = \sum_{j=0}^{\frac{n}{2}-1} f_{2j+1} x^{2j}$ then

- $f(\omega^p) = \sum_{j=0}^{\frac{n}{2}-1} f_{2j}\omega^{2jp} + \omega^p \sum_{j=0}^{\frac{n}{2}-1} f_{2j+1}\omega^{2jp} =$

$$= R_0(\omega) + \omega^p R_1(\omega)$$

- $f(\omega^{p+\frac{n}{2}}) = \sum_{j=0}^{\frac{n}{2}-1} f_{2j}\omega^{2j(p+\frac{n}{2})} + \sum_{j=0}^{\frac{n}{2}-1} f_{2j+1}\omega^{(2j+1)(p+\frac{n}{2})} =$

$$= \sum_{j=0}^{\frac{n}{2}-1} f_{2j}\omega^{2jp} - \omega^p \sum_{j=0}^{\frac{n}{2}-1} f_{2j+1}\omega^{2jp} = R_0(\omega) - \omega^p R_1(\omega)$$

FFT,Cooley and Tukey (1965)

Since

- $f(\omega^p) = R_0(\omega) + \omega^p R_1(\omega)$
- $f(\omega^{p+\frac{n}{2}}) = R_0(\omega) - \omega^p R_1(\omega)$

Then

$$DFT_\omega(f) = (f(1), \ldots, f(\omega^{n-1})) =$$

$$= (R_0 + R_1, R_0 + \omega R_1, \ldots, R_0 + \omega^{\frac{n}{2}-1} R_1,$$

$$R_0 - R_1, R_0 - \omega R_1, \ldots, R_0 - \omega^{\frac{n}{2}-1} R_1)$$

FFT,Cooley and Tukey (1965)

In[8]:=

```
FFT[R_, w_, n_] := Module[{RR, a, m, R0, R1, L0, L1},
  RR = PadRight[R, n];
  If[n == 1, Return[{RR[[1]]}]];
  m = n / 2;
  a = w * w;
  R0 = RR[[1 ;;  ;; 2]];
  R1 = RR[[2 ;;  ;; 2]];
  Print[L0 = FFT[R0, a, m]];
  Print[L1 = FFT[R1, a, m]];
  Print[L1 = w^Range[0, m - 1] * L1];
  Return[Catenate[{L0 + L1, L0 - L1}]]]
```

In[9]:=

Homework (Fast Mul. of Pol. via FFT)

Now $\deg(f), \deg(g) < n = 2^k$ and $\deg(f \cdot g) < n$

1. Compute $\gamma = FFT_\omega(f) \cdot FFT_\omega(g)$.
2. $f \cdot g = FFT_\omega^{-1}(\gamma) = \frac{1}{n} FFT_{\omega^{-1}}(\gamma)$

a) Write a code PFFT[P, Q, n] which returns P Q by using the code FFT of exercise 2. The input should be polynomials P and Q, and the output should be the polynomial P Q.

b) Compare the results and the timing of computing these operations with those of Karatsuba. Use again big random polynomials for doing this test.