Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
M. Ahmadi, A. R. Molla, and O. Saukh, November 10, 2015

Algorithm Theory, Winter Term 2015/16
Problem Set 2 - Sample Solution

Exercise 1: Multiplication of Polynomials (2 + 7 points)

Given a polynomial $p(x)$:

$$p(x) = x^3 - x^2 + 2x + 1$$

- Compute $DFT_4(a)$, where a is the coefficient vector of polynomial p.
- Compute the coefficient representation of $p(x)^2$ by using the FFT algorithm from the lecture.

Remark: Instead of using exact numbers for the point-wise evaluations (which involves irrational numbers) you can also round numbers to, say, 3+ digits after the decimal point. This also reflects what happens in an implemented version of FFT, as exact algebraic evaluations would not lead to an $O(n \log n)$ running time. Those unfamiliar with complex numbers should ask fellow students for some help - calculating roots of unity and multiplying two complex numbers is all you need for this exercise.

Solution

DFT polynomial representation

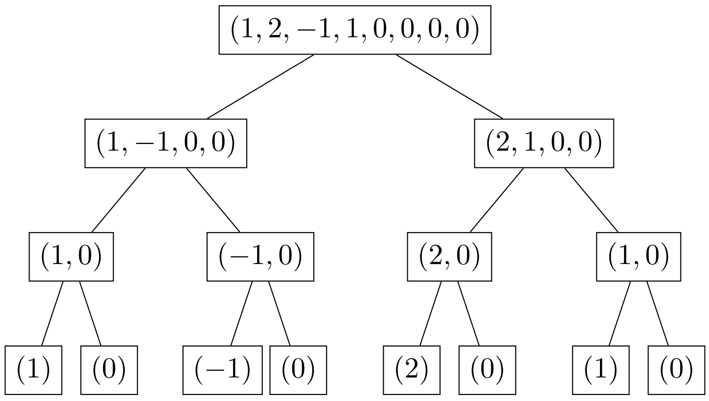
Consider the coefficient vector a in the reverse order i.e., $a = (1, 2, -1, 1)$. It is known that $DFT_4(a) = \big(p(\omega_4^0), p(\omega_4^1), p(\omega_4^2), p(\omega_4^3)\big)^T$ is a vector containing the values of polynomial in each of four roots of unity. The roots of unity are: $\omega_4^0 = 1$, $\omega_4^1 = i$, $\omega_4^2 = -1$, $\omega_4^3 = -i$. So, if we compute polynomial in each of those points, we get: $DFT_4(a) = (3, 2+i, -3, 2-i)$

FFT Plynomial multiplication

To multiply two polynomials p and p' of degree d_p and $d_{p'}$ respectively, by using interpolation, we must evaluate those polynomials at $d_p + d_{p'} + 1$ distinct points. In our case we have two identical polynomials of degree 3, i.e., 7 points are needed. To use FFT, rounding up to the next power of 2 is advised, as it simplifies the computation by more than the additional amount of work we need to do. Thus, from now on, $N = 8$.
The coefficient vector for our polynomial p is $\boldsymbol{a} = (1, 2, -1, 1)$. For the upcoming multiplication of p with itself we will calculate $DFT_N(\boldsymbol{a})$ which is the abbreviation for $\big(p(\omega_8^0), \ldots, p(\omega_8^7)\big)$, i.e., the evaluation of p at the 8 roots of unity. The polynomial p^2 has a descriptive vector of length 7, but the FFT algorithm is much clearer if all polynomials use the same length of their descriptive vector, and it should be of length N, thus we redefine $\boldsymbol{a} := (1, 2, -1, 1, 0, 0, 0, 0)$.

Divide: First we need to split p into p_0 and p_1. Their respective vectors are $\boldsymbol{a_0} := (1,-1,0,0)$ and $\boldsymbol{a_1} := (2,1,0,0)$. Those we split further until we reach a descriptive vector of length 1: $\boldsymbol{a_{00}} := (1,0), \boldsymbol{a_{01}} := (-1,0), \boldsymbol{a_{10}} := (2,0), \boldsymbol{a_{11}} := (1,0)$ and $\boldsymbol{a_{000}} := (1), \boldsymbol{a_{001}} := (0), \boldsymbol{a_{010}} := (-1), \boldsymbol{a_{011}} := (0), \boldsymbol{a_{100}} := (2), \boldsymbol{a_{101}} := (0), \boldsymbol{a_{110}} := (1), \boldsymbol{a_{111}} := (0)$. Now, a polynomial of degree 0 is easy to calculate, as it represents a constant function: $\mathrm{DFT}_1(\boldsymbol{a_{000}}) = p_{000}(\omega_1^0) = 1$. As you can see, we could already have stopped at polynomials $p_{00}, \ldots, p_{11}$, because they already represented polynomials of degree 0. But let's be meticulous:

k	ω_1^k	p_{000}	p_{001}	p_{010}	p_{011}	p_{100}	p_{101}	p_{110}	p_{111}
0	1	1	0	-1	0	2	0	1	0



Conquer: Remember that our goal in each step is to combine two polynomials of degree $n/2-1$ to one polynomial of degree $n-1$, but in the point-value representation, i.e., we need to get the values of that polynomial at n different roots of unity. The formula for combination is

$$p(\omega_n^k) = p_0(\omega_{n/2}^{k \mod n/2}) + \omega_n^k \cdot p_1(\omega_{n/2}^{k \mod n/2}).$$

E.g., $p_{10}(\omega_2^1) = p_{10}(-1) = p_{100}(\omega_1^{1 \mod 1}) + \omega_2^1 p_{101}(\omega_1^{1 \mod 1}) = 2 + (-1) \cdot 0 = 2$.

k	ω_2^k	p_{00}	p_{01}	p_{10}	p_{11}
0	1	1	-1	2	1
1	-1	1	-1	2	1

The next step is slightly more complicated. E.g., $p_1(\omega_4^3) = p_1(-i) = p_{10}(\omega_2^{3 \mod 2}) + \omega_4^3 p_{11}(\omega_2^{3 \mod 2}) = p_{10}(-1) + (-i)p_{11}(-1) = 2 - i$

k	ω_4^k	p_0	p_1
0	1	$1 + 1 \cdot (-1)$	$2 + 1 \cdot 1$
1	i	$1 + i \cdot (-1)$	$2 + i \cdot 1$
2	-1	$1 - 1 \cdot (-1)$	$2 - 1 \cdot 1$
3	$-i$	$1 - i \cdot (-1)$	$2 - i \cdot 1$

Or more nicely:

k	ω_4^k	p_0	p_1
0	1	0	3
1	i	$1 - i$	$2 + i$
2	-1	2	1
3	$-i$	$1 + i$	$2 - i$

The last step involves 4 new roots of unity of the form $\frac{\pm 1 \pm i}{\sqrt{2}}$, which makes calculation a bit more difficult.
Example calculation: $p(\omega_8^5) = p(\frac{-1-i}{\sqrt{2}}) = p_0(\omega_4^{5 \mod 4}) + \omega_8^5 p_1(\omega_4^{5 \mod 4}) = p_0(i) + \frac{-1-i}{\sqrt{2}} p_1(i) = 1 - i + \frac{-1-i}{\sqrt{2}}(2+i) = 1 - i + \frac{1}{\sqrt{2}}(\sqrt{2} + \sqrt{2}i - 2 - i - 2i + 1) = \frac{1}{\sqrt{2}}(-1 + \sqrt{2} + (-3 - \sqrt{2})i)$

k	ω_8^k	$p(computation)$	$p(result)$
0	1	$0+1\cdot 3$	3
1	$\frac{1+i}{\sqrt{2}}$	$1-i+\frac{1+i}{\sqrt{2}}(2+i)$	$\frac{1}{\sqrt{2}}(1+\sqrt{2}+(3-\sqrt{2})i)$
2	i	$2+i\cdot 1$	$2+i$
3	$\frac{-1+i}{\sqrt{2}}$	$1+i+\frac{-1+i}{\sqrt{2}}(2-i)$	$\frac{1}{\sqrt{2}}(-1+\sqrt{2}+(3+\sqrt{2})i)$
4	-1	$0-1\cdot 3$	-3
5	$\frac{-1-i}{\sqrt{2}}$	$1-i+\frac{-1-i}{\sqrt{2}}(2+i)$	$\frac{1}{\sqrt{2}}(-1+\sqrt{2}+(-3-\sqrt{2})i)$
6	$-i$	$2-i\cdot 1$	$2-i$
7	$\frac{1-i}{\sqrt{2}}$	$1+i+\frac{1-i}{\sqrt{2}}(2-i)$	$\frac{1}{\sqrt{2}}(1+\sqrt{2}+(-3+\sqrt{2})i)$

Notice that each table contains about N entries. Calculating those entries can be done in constant time as long as we are using floating point arithmetic.

Multiplying the polynomials

Now comes the easiest part. Understand that we just changed the *representation* of the polynomial p from its coefficient vector $\boldsymbol{a} = (1, 2, -1, 1)$ to the representation:

$$DFT_8(\boldsymbol{a}) = (3, \frac{1}{\sqrt{2}}(1+\sqrt{2}+(3-\sqrt{2})i), 2+i, \frac{1}{\sqrt{2}}(-1+\sqrt{2}+(3+\sqrt{2})i), -3, \frac{1}{\sqrt{2}}(-1+\sqrt{2}+(-3-\sqrt{2})i), 2-i, \frac{1}{\sqrt{2}}(1+\sqrt{2}+(-3+\sqrt{2})i)), \quad (1)$$

which is an abbreviation of the point value representation

$$p \equiv \{(\omega_8^0, 3), (\omega_8^1, \frac{1}{\sqrt{2}}(1+\sqrt{2}+(3-\sqrt{2})i)), (\omega_8^2, 2+i), (\omega_8^3, \frac{1}{\sqrt{2}}(-1+\sqrt{2}+(3+\sqrt{2})i)), (\omega_8^4, -3), (\omega_8^5, \frac{1}{\sqrt{2}}(-1+\sqrt{2}+(-3-\sqrt{2})i)), (\omega_8^6, 2-i), (\omega_8^7, \frac{1}{\sqrt{2}}(1+\sqrt{2}+(-3+\sqrt{2})i))\}. \quad (2)$$

(the DFT_N part only says in a short way which points $x_0, \ldots, x_{N-1}$ are used for the evaluation.)
We learned in the lecture that one can multiply polynomials quickly in this representation by just multiplying the evaluated points with each other. So lets do that:

x	$p(x)$	$p^2(x) =: q$
ω_8^0	3	9
ω_8^1	$\frac{1}{\sqrt{2}}(1+\sqrt{2}+(3-\sqrt{2})i)$	$-4+4\sqrt{2}+(1+2\sqrt{2})i$
ω_8^2	$2+i$	$3+4i$
ω_8^3	$\frac{1}{\sqrt{2}}(-1+\sqrt{2}+(3+\sqrt{2})i)$	$-4-4\sqrt{2}+(-1+2\sqrt{2})i$
ω_8^4	-3	9
ω_8^5	$\frac{1}{\sqrt{2}}(-1+\sqrt{2}+(-3-\sqrt{2})i)$	$-4-4\sqrt{2}+(1-2\sqrt{2})i$
ω_8^6	$2-i$	$3-4i$
ω_8^7	$\frac{1}{\sqrt{2}}(1+\sqrt{2}+(-3+\sqrt{2})i)$	$-4+4\sqrt{2}+(-1-2\sqrt{2})i$

and in direct point-value representation:

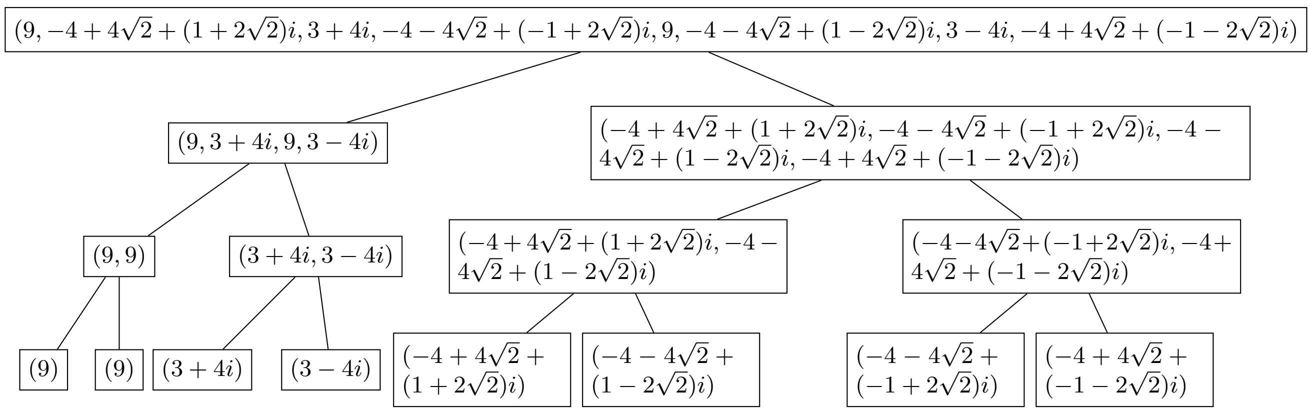
$$p^2 \equiv \{(\omega_8^0, 9), (\omega_8^1, -4+4\sqrt{2}+(1+2\sqrt{2})i), (\omega_8^2, 3+4i), (\omega_8^3, -4-4\sqrt{2}+(-1+2\sqrt{2})i), (\omega_8^4, 9), (\omega_8^5, -4-4\sqrt{2}+(1-2\sqrt{2})i), (\omega_8^6, 3-4i), (\omega_8^7, -4+4\sqrt{2}+(-1-2\sqrt{2})i)\} \quad (3)$$

Unfortunately for us the point value representation of p^2 is hard to grasp nor very handy for various tasks (other than multiplying polynomials), that's why we want it transformed back into the vector form.

Inverse DFT

To get the descriptive vector $\boldsymbol{b}$ of $p^2 =: q$ we have to solve a linear equation of the form $W \cdot \boldsymbol{b} = \boldsymbol{y}$, where $y_i = q(\omega_N^i)$. We did see how W^{-1} looks like and we realized that there is a polynomial $r(x) := y_0 + y_1 x + \cdots + y_{N-1}x^{N-1}$, which we simply have to evaluate at the same points $\omega_8^0, \ldots \omega_8^7$ to get the values $b'_0, b'_1, \ldots, b'_{N-1}$ which we can transform into the vector $\boldsymbol{b}$.
We again use FFT to get $\boldsymbol{b'} := \mathrm{DFT}_N(\boldsymbol{y})$, which we can easily transform into $\boldsymbol{b}$ by changing the order a bit and by dividing all results by N.

Divide:


$(9, -4+4\sqrt{2}+(1+2\sqrt{2})i, 3+4i, -4-4\sqrt{2}+(-1+2\sqrt{2})i, 9, -4-4\sqrt{2}+(1-2\sqrt{2})i, 3-4i, -4+4\sqrt{2}+(-1-2\sqrt{2})i)$
$(9, 3+4i, 9, 3-4i)$
$(-4+4\sqrt{2}+(1+2\sqrt{2})i, -4-4\sqrt{2}+(-1+2\sqrt{2})i, -4-4\sqrt{2}+(1-2\sqrt{2})i, -4+4\sqrt{2}+(-1-2\sqrt{2})i)$
$(9, 9)$
$(3+4i, 3-4i)$
$(-4+4\sqrt{2}+(1+2\sqrt{2})i, -4-4\sqrt{2}+(1-2\sqrt{2})i)$
$(-4-4\sqrt{2}+(-1+2\sqrt{2})i, -4+4\sqrt{2}+(-1-2\sqrt{2})i)$
(9)
(9)
$(3+4i)$
$(3-4i)$
$(-4+4\sqrt{2}+(1+2\sqrt{2})i)$
$(-4-4\sqrt{2}+(1-2\sqrt{2})i)$
$(-4-4\sqrt{2}+(-1+2\sqrt{2})i)$
$(-4+4\sqrt{2}+(-1-2\sqrt{2})i)$

Conquer: And again, the formula for combination is

$$q(\omega_n^k) = q_0(\omega_{n/2}^{k \bmod n/2}) + \omega_n^k q_1(\omega_{n/2}^{k \bmod n/2}).$$

E.g., $r_{10}(\omega_2^1) = r_{10}(-1) = r_{100}(\omega_1^{1 \bmod 1}) + \omega_2^1 r_{101}(\omega_1^{1 \bmod 1}) = -4+4\sqrt{2}+(1+2\sqrt{2})i + (-1)\cdot(-4-4\sqrt{2}+(1-2\sqrt{2})i) = -8+2i$.

k	ω_2^k	r_{00}	r_{01}	r_{10}	r_{11}
0	1	18	6	$-8+2i$	$-8-2i$
1	-1	0	$8i$	$8\sqrt{2}+4\sqrt{2}i$	$-8\sqrt{2}+4\sqrt{2}i$

Example for step 2: $r_1(\omega_4^3) = r_1(-i) = r_{10}(\omega_2^{3 \bmod 2}) + \omega_4^3 r_{11}(\omega_2^{3 \bmod 2}) = r_{10}(-1) + (-i)r_{11}(-1) = 12\sqrt{2}+12\sqrt{2}i$

k	ω_4^k	r_0	r_1	$r_0(result)$	$r_1(result)$
0	1	$18+1\cdot 6$	$-8+2i+1\cdot(-8-2i)$	24	-16
1	i	$0+i\cdot(8i)$	$8\sqrt{2}+4\sqrt{2}i+i\cdot(-8\sqrt{2}+4\sqrt{2}i)$	-8	$4\sqrt{2}-4\sqrt{2}i$
2	-1	$18+(-1)\cdot(8)$	$-8+2i+(-1)\cdot(-8-2i)$	12	$4i$
3	$-i$	$0+(-i)\cdot(8i)$	$8\sqrt{2}+4\sqrt{2}i+(-i)\cdot(-8\sqrt{2}+4\sqrt{2}i)$	8	$12\sqrt{2}+12\sqrt{2}i$

And the last step:

k	ω_8^k	r	$r(result)$
0	1	$24 + 1 \cdot (-16)$	8
1	$\frac{1+i}{\sqrt{2}}$	$-8 + \frac{1+i}{\sqrt{2}}(4-4i)\sqrt{2}$	0
2	i	$12 + i \cdot 4i$	8
3	$\frac{-1+i}{\sqrt{2}}$	$8 + \frac{-1+i}{\sqrt{2}}(12+12i)\sqrt{2}$	-16
4	-1	$24 + (-1) \cdot (-16)$	40
5	$\frac{-1-i}{\sqrt{2}}$	$-8 + \frac{-1-i}{\sqrt{2}}(4-4i)\sqrt{2}$	-16
6	$-i$	$12 + (-i) \cdot (4i)$	16
7	$\frac{1-i}{\sqrt{2}}$	$8 + \frac{1-i}{\sqrt{2}}(12+12i)\sqrt{2}$	32

Thus, $\boldsymbol{b'} = DFT_8(\boldsymbol{y}) = (8, 0, 8, -16, 40, -16, 16, 32)$. To get $\boldsymbol{b}$ we have to divide all results by $N = 8$ and reverse the order of all elements, except the first one: $\boldsymbol{b} = \frac{1}{N}(b'_0, b'_{N-1}, b'_{N-2}, \ldots, b'_2, b'_1) = (1, 4, 2, -2, 5, -2, 1, 0)$.
The final result is:

$$q(x) = p^2(x) = x^6 - 2x^5 + 5x^4 - 2x^3 + 2x^2 + 4x + 1$$

Exercise 2: Polynomial to the power of k (3 points)

Given a polynomial $p(x)$ of degree n and an integer $k \geq 2$, the goal of this problem is to compute the k^{th} power $p^k(x)$ of $p(x)$ in an efficient way. For simplicity, we assume that k is a power of 2, that is, $k = 2^\ell$ for some integer $\ell \geq 1$.

- Describe an efficient algorithm to compute $p^k(x)$ polynomial using the *Fast Polynomial Multiplication* algorithm from the lecture.
- What is the asymptotic runtime of your algorithm in terms of k and n? Explain your answer.

Solution 2

We compute the k^{th} power of the polynomial $p(x)$ by iterative multiplication, as

$$p(x)^k = (\ldots((p(x)^2)^2 \ldots)^2$$

where square is taken $\log k$ times (assuming k is a power of 2).
We know that using the FFT algorithm from the lecture, two polynomials of degree n can be multiplied in $O(n \log n)$ time. Notice that on every iterative step i of the algorithm, we need to multiply two polynomials of degree $(2^{i-1} \cdot n)$ and get a polynomial of degree $(2^i \cdot n)$. Now to compute $p(x)^k$ (where $k = 2^\ell$ for some ℓ), there will be $\log k$ such iterations.
Therefore, the asymptotic running time of the algorithm (i.e., iterative multiplication of two same degree polynomials) would be:

$$\sum_{i=0}^{\log k - 1} c \cdot 2^i \cdot n \cdot \log(2^i \cdot n) \qquad \text{[where } c \text{ is some constant in } O(n \log n) \text{ bound]}$$

Let us compute this sum:

$$\begin{aligned}
&\sum_{i=0}^{\log k - 1} c \cdot 2^i \cdot n \cdot \log(2^i \cdot n) = cn \cdot \sum_{i=0}^{\log k - 1} 2^i \cdot (i + \log(n)) \\
&= cn \cdot \sum_{i=0}^{\log k - 1} i \cdot 2^i + cn \cdot \log(n) \cdot \sum_{i=0}^{\log k - 1} 2^i \\
&\leq cn \cdot \left((\log(k) - 1) \cdot 2^{\log(k)}\right) + cn \cdot \log(n) \cdot \left(2^{\log(k)} - 1\right) \\
&\leq cnk \cdot \log(nk)
\end{aligned}$$

Hence, the asymptotic running time of the algorithm is $O(nk \cdot \log(nk))$.