Junior-Prof. Dr. Robert Elsässer, Marco Muñiz, Phillip Heidegger WS 2009/2010

Algorithms Theory, Solution for Assignment 2

http://lak.informatik.uni-freiburg.de/lak_teaching/ws09_10/algo0910.php

Exercise 2.1 - Fast Fourier Transform

The two polynomials have degree less than 2, hence the polynomial pq has degree less than 4. We represent p and q by four entries in an array and compute DFT_4 using FFT.

$$p(x) = 3x + 1 \qquad p_a = [1, 3, 0, 0]$$
$$q(x) = 7x + 4 \qquad q_a = [4, 7, 0, 0]$$

We split p_a and q_a into two parts:

$$p_a^1 = [1, 0] \qquad p_a^2 = [3, 0]$$
$$q_a^1 = [4, 0] \qquad q_a^2 = [7, 0]$$

For $1 \leq k \leq 3$, it holds that:

$$DFT_k(p_a, 4) = (DFT(p_a^1, 2), DFT(p_a^1, 2))_k + \omega_4^k (DFT(p_a^2, 2), DFT(p_a^2, 2))_k \tag{1}$$

Hint: If v_1 is a vector with n elements and v_2 is a vector with m elements, then (v_1, v_2) is a vector with $n + m$ elements, Then:

$$(v_1, v_2)_k := \begin{cases} v_k & \text{if } 1 \leq k \leq n \\ v_j & \text{if } n < k \leq n + m \text{ and } j = k - n \end{cases}$$

Example The following example demonstrates the notation given above:

$$((1, 2, 3), (4, 5)) = (1, 2, 3, 4, 5)$$

Hence, $(DFT(p_a^1, 2), DFT(p_a^1, 2))$ is a vector with 4 entries, and the first two entries are the same as the second one.

We state

$$\omega_4^0 = 1 \qquad \omega_4^1 = i$$
$$\omega_4^2 = -1 \qquad \omega_4^3 = -i$$

Hence we can write $DFT(p_a, 4) = FFT(p_a, 4)$ as:

$$\begin{aligned} FFT(p_a, 4) = (&FFT([1, 0], 2)_1 + 1 \cdot FFT([3, 0], 2)_1, \\ &FFT([1, 0], 2)_2 + i \cdot FFT([3, 0], 2)_2, \\ &FFT([1, 0], 2)_1 + (-1) \cdot FFT([3, 0], 2)_1, \\ &FFT([1, 0], 2)_2 + (-i) \cdot FFT([3, 0], 2)_2 \qquad) \end{aligned}$$

Now we have to compute $FFT([1, 0], 2)$ and $FFT([3, 0], 2)$.

1. First we compute $FFT([1, 0], 2)$. It is defined as:

$$\begin{aligned} FFT([1, 0], 2) &= ((FFT([1], 1), FFT([1], 1))_1 + 1 \cdot (FFT([0], 1), FFT([0], 1))_1, \\ &\qquad (FFT([1], 1), FFT([1], 1))_2 + (-1) \cdot (FFT([0], 1), FFT([0], 1))_2 \qquad) \\ &= (1 + 0, 1 - 0) = (1, 1) \end{aligned}$$

2. Now, $FFT([3,0],2)$ yields:

$$FFT([3,0],2) = (3,3)$$

We optain

$$FFT(p_a,4) = (1+3, 1+3i, 1-3, 1-3i) = (4, 1+3i, -2, 1-3i) \tag{2}$$

For q_a it holds that:

$$\begin{aligned} FFT(q_a,4) = (&FFT([4,0],2)_1 + 1 \cdot FFT([7,0],2)_1, \\ &FFT([4,0],2)_2 + i \cdot FFT([7,0],2)_2, \\ &FFT([4,0],2)_1 + (-1) \cdot FFT([7,0],2)_1, \\ &FFT([4,0],2)_2 + (-i) \cdot FFT([7,0],2)_2 \quad) \end{aligned}$$

1. First we compute $FFT([4,0],2)$.

$$FFT([4,0],2) = (4,4)$$

2. Now, $FFT([7,0],2)$ yields:

$$FFT([7,0],2) = (7,7)$$

Then,

$$FFT(q_a,4) = (4+7, 4+7i, 4-7, 4-7i) = (11, 4+7i, -3, 4-7i) \tag{3}$$

Hence we get the result for $p \cdot q$ by multiplying (2) and (3) :

$$\begin{aligned} FFT(p \cdot q, 4) &= (4 \cdot 11, (1+3i) \cdot (4+7i), -2 \cdot (-3), (1-3i) \cdot (4-7i)) \\ &= (44, -17+19i, 6, -17-19i) \end{aligned}$$

This yields

$$\begin{aligned} pq(\omega_4^0) &= pq(1) = 44 \\ pq(\omega_4^1) &= pq(i) = -17+19i \\ pq(\omega_4^2) &= pq(-1) = 6 \\ pq(\omega_4^3) &= pq(-i) = -17-19i \end{aligned}$$

Hence we have a point-value representation of pq.

Interpolation

To compute the coefficients we set $r(x) := [44, -17+19i, 6, -17-19i]$. We compute FFT(r,4) by first splitting r into two parts: $r^1 = [44,6]$ and $r^2 = [-17+19i, -17-19i]$.

$$\begin{aligned} FFT(r,4) = (&FFT([44,6],2)_1 + FFT([-17+19i, -17-19i],2)_1, \\ &FFT([44,6],2)_2 + iFFT([-17+19i, -17-19i],2)_2, \\ &FFT([44,6],2)_1 - FFT([-17+19i, -17-19i],2)_1, \\ &FFT([44,6],2)_2 - iFFT([-17+19i, -17-19i],2)_2 \quad) \end{aligned}$$

We compute FFT([44,6],2) and FFT([-17 + 19i, -17 - 19i],2):

$$\begin{aligned} FFT([44,6],2) &= (44+6, 44-6) = (50,38) \\ FFT([-17+19i, -17-19i],2) &= (-34, 38i) \end{aligned}$$

Hence:

$$FFT(r,4) = (50-34, 38+38i^2, 50+34, 38-38i^2) = (16,0,84,76)$$

From this we obtain the coefficients

$$a_0 = \frac{1}{4} \cdot 16 = 4 \qquad a_1 = \frac{1}{4} \cdot 76 = 19$$
$$a_2 = \frac{1}{4} \cdot 84 = 21 \qquad a_3 = \frac{1}{4} \cdot 0 = 0$$

and hence

$$pq = 0x^3 + 21x^2 + 19x + 4$$

Exercise 2.2 - FFT

1. Define

$$p_A = a_{m-1}x^{m-1} + \cdots a_1 x + a_0$$
$$p_B = b_{m-1}x^{m-1} + \cdots b_1 x + b_0$$

for $0 \leq j \leq m-1$ where

$$a_j = \begin{cases} 1 & \text{if } j \in A \\ 0 & \text{if } j \notin A \end{cases} \qquad b_j = \begin{cases} 1 & \text{if } j \in B \\ 0 & \text{if } j \notin B \end{cases}$$

The polynomial $p_C = p_A \cdot p_B = k_{2m-2}x^{2m-2} + \cdots k_1 x + k_0$ represents the set $C = A + B$. For $0 \leq j \leq 2m-1$ it holds that

$$j \in C \Leftrightarrow k_j > 0$$

Since p_c can be computed by FFT in time $\mathcal{O}(m \ \log m)$, the statement holds.

2. The numbers k_j are the solution for the second question. Please notice that it is important to choose $a_j = 1$ if $j \in A$ and $b_j = 1$ if $j \in B$.

3. In this part we need to count for all x all pairs $(a, b) \in A \times B$, such that there exists a $c \in \mathbb{N}$ with $x = c \cdot (a + b)$. First assume we have a fixed x.

Assume for example $x = 6$. Computing d_6 can by done by summing up k_1, k_2, k_3 and k_6. For $x = 8$ we sum up k_1, k_2, k_4, k_8.

More generally, for each $x \in \{1, \ldots, 2m-2\}$:

$$d_x = \sum_{i=1, i|x}^{2m-2} k_i$$

We can write this into a table:

$$\begin{array}{lclllllll} d_1 & = & k_1 & & & & & \\ d_2 & = & k_1 & +k_2 & & & & \\ d_3 & = & k_1 & & +k_3 & & & \\ d_4 & = & k_1 & +k_2 & & +k_4 & & \\ d_5 & = & k_1 & & & & +k_5 & \\ d_6 & = & k_1 & +k_2 & +k_3 & & & +k_6 \end{array}$$

It's easy to see that k_1 is part of each sum, while k_2 is part of $d_2, d_4, d_6, \ldots, d_{2m-2}$. In general, for each $i \in \{1, \ldots, 2m-2\}$ the value k_i is part of $d_i, d_{2i}, d_{3i}, \ldots, d_{ki}$, where

$$k \cdot i \leq 2m - 2 < r(k+1) \cdot i$$

Our algorithm takes k_j as input. It computes for each $x \in \{1, \ldots, 2m-2\}$ the number d_x.

```
INPUT: k[]
d[] = new Array[1..2m−2](0);
for each i in [1..2m−2] do
  for (x = i; x+ = i; x < 2m − 2)
    d[x] = d[x] + k[i]
OUTPUT: d[]
```

For $n = 2m - 2$, the runtime of the algorithm $T(m)$ is bounded by:

$$\begin{aligned} T(m) &\leq \sum_{i=1}^{n} \left(\frac{\sum_{x=i}^{n} 1}{i} \right) \\ &= \sum_{i=1}^{n} \left(\frac{n-i}{i} \right) \\ &\leq \sum_{i=1}^{n} \frac{n}{i} \\ &= n \left(1 + \sum_{i=2}^{n} \frac{1}{i} \right) \\ &\leq n \cdot (1 + \log n) \in \mathcal{O}(n \log n) \end{aligned}$$

Hence, the runtime is in $\mathcal{O}(m \log m)$.

Exercise 2.3 - Randomized Quicksort

1. $T(n) = \Theta(n^2)$ arises when the worst-case partitioning occurs (i.d. partitioning yields two sub-problems, with number of elements $n-1$ and 0 respectively).

   Possible permutations π of n and probabilities for p_l and p_r are:

   - $\pi = n_1, n_2, \ldots, n_m$ and $p_l = 0, p_r = 1$.
   - Symmetrically we have: $\pi = n_m, n_{m-1}, \ldots, n_1$ and $p_l = 1, p_r = 0$.
   - $\pi = n_1, n_2, \ldots, n_m$ and $p_l = 0.5, p_r = 0.5$. One possible execution of Randomized Quicksort could lead to the following partitions:

left	right	
$\emptyset$	$n_2, n_3, \ldots, n_m$	$pivot = l = n_1$
$n_2, n_3 \ldots, n_{m-1}$	$\emptyset$	$pivot = r = n_m$
$\emptyset$	$n_3, \ldots, n_{m-1}$	$pivot = l = n_2$
$\vdots$		

2. We prove that $T(n) \in \mathcal{O}(n \log n)$.

   We choose a constant c_1, such that $\forall i \in \{1, \ldots, n-1\}$

$$T(i) \leq c_1 \cdot i \log i \quad .$$

   and we prove for large n that $T(n) \leq c_1 n \log n$.

The definition of $\Theta(n)$ and $T(n)$ states that for some $c \in \mathbb{N}$:

$$
\begin{aligned}
T(n) &\leq \frac{2}{n}\sum_{k=1}^{n-1} T(k) + cn \\
&\leq \frac{2}{n}\sum_{k=1}^{n-1} c_1 k \log k + cn \\
&= \frac{2c_1}{n}\left(\sum_{k=1}^{n/2} k \log k + \sum_{k=n/2+1}^{n-1} k \log k\right) + cn
\end{aligned}
$$

Since log is a monotone increasing function

$$
= \frac{2c_1}{n}\left(\sum_{k=1}^{n/2} k \log \frac{n}{2} + \sum_{k=n/2+1}^{n-1} k \log n\right) + cn
$$

We use $\log n/2 = \log n - \log 2$ and $\log 2 \geq 1$

$$
\begin{aligned}
&\leq \frac{2c_1}{n}\left((\log n - 1)\sum_{k=1}^{n/2} k + \log n \sum_{k=n/2+1}^{n-1} k\right) + cn \\
&= \frac{2c_1}{n}\left(\log n \sum_{k=1}^{n-1} k - \sum_{k=1}^{n/2} k\right) + cn \\
&= \frac{2c_1}{n}\left(\log n \frac{(n-1)(n-2)}{2} - \frac{(n/2-1)(n/2-2)}{2}\right) + cn \\
&= \frac{c_1}{n}\left(\log n(n-1)(n-2) - \frac{1}{4}(n-2)(n-4)\right) + cn \\
&\leq c_1 n \log n - \frac{c_1}{4n}(n^2 - 6n + 8) + cn \\
&= c_1 n \log n - \frac{c_1 n}{4} + \frac{3}{2}c_1 - \frac{2c_1}{n} + cn \\
&\leq c_1 n \log n - n\frac{c_1}{4} + \frac{3}{2}c_1 + cn
\end{aligned}
$$

We choose $c_1 = 4\left(c + \frac{3}{2}\right)$

$$
\leq c_1 n \log n - cn + cn - \frac{3}{2}n + \frac{3}{2}c_1
$$

For large n it holds $n > c_1$, which yields $\frac{3}{2}c_1 \leq \frac{3}{2}n$.

$$
\leq c_1 n \log n
$$

Exercise 2.4 - RSA

1. Given, $p = 19, q = 29$ and $e = 5$. Compute $n = pq = 551$. Use the extended−Euclid algorithm with $a = (p-1)(q-1) = 504$ and $b = e = 5$ to compute d as the multiplicative inverse of e modulo $(p-1)(q-1)$.

input	output
$(504, 5)$	$(1, -1, 1 - \lfloor \frac{504}{5} \rfloor \cdot -1) = (1, -1, 101)$
$(5, 4)$	$(1, 1, 0 - \lfloor \frac{5}{4} \rfloor \cdot 1) = (1, 1, -1)$
$(4, 1)$	$(1, 0, 1 - \lfloor \frac{4}{1} \rfloor \cdot 0) = (1, 0, 1)$
$(1, 0)$	$(1, 1, 0)$

The extended−Euclid algorithm returns the modular multiplicative inverses such that

$$\gcd(a, b) = ax + by$$
$$1 = 504 \cdot (-1) + 5 \cdot (101)$$

Since $d * e \mod 504 = 1$, we have $d = y = 101$.
Public key $P = (e, n) = (5, 551)$, secret key $S = (d, n) = (101, 551)$.

2. $P(M) = P(22) = 22^5 \mod 551 = 129$