Devoir 10 IFT2125-A-H19

Student: Qiang Ye (20139927)

Date: 29 March 2019

mailto: samuel.ducharme@umontreal.ca (mailto:samuel.ducharme@umontreal.ca)

Question & Answer

Utiliser l'algorithme FFT pour calculer le produits suivant de deux polynômes

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

en montrant dans les tableaux prévu toutes les itérations (utilisez ce qu'il faut, il se peut qu'il reste de la place inutilisée).

Mettez le résultat final - le polynôme obtenu - dans la boîte ci-dessous :

$$\boxed{x^6 - 4x^4 + 3x^3 + 3x^2 - 7x + 2}$$

Montrez ici votre travail:

1. Levecteur Ω

1	$\frac{1+i}{\sqrt{2}}$	i	$\frac{-1+i}{\sqrt{2}}$	-1	$\frac{-1-i}{\sqrt{2}}$	$-i$	$\frac{1-i}{\sqrt{2}}$

2. Les vecteurs initiaux:

1	0	0	0	−3	0	1	0

2	0	0	0	−1	0	1	0

3. Les itérations (transformées successives):

1	1	0	0	−3	−3	1	1

1	1	1	1	-2	$-3+i$	-4	$-3-i$

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

2	2	0	0	-1	-1	1	1

2	2	2	2	0	$-1+i$	-2	$-1-i$

2	$2-\sqrt{2}$	$2-2i$	$2+\sqrt{2}$	2	$2+\sqrt{2}$	$2+2i$	$2-\sqrt{2}$

4. La multiplication - la transformée du résultat (obtenue à partir des tranformées de la page précédente):

-2	$(6-5\sqrt{2}) + (2 - 2\sqrt{2})i$	$-6-10i$	$(6+5\sqrt{2}) - (2 + 2\sqrt{2})i$	6	$(6+5\sqrt{2}) + (2 + 2\sqrt{2})i$	$-6+10i$	$(6-5\sqrt{2}) + (2\sqrt{2} - 2)i$

5. La transformée inverse:

Ω:

1	$\frac{1-i}{\sqrt{2}}$	$-i$	$\frac{-1-i}{\sqrt{2}}$	-1	$\frac{-1+i}{\sqrt{2}}$	i	$\frac{1+i}{\sqrt{2}}$

Itération:

(Fisrt line is just the re-ordered Multiplication)

-2	6	$-6-10i$	$-6+10i$	$(6-5\sqrt{2})+(2-2\sqrt{2})i$	$(6+5\sqrt{2})+(2+2\sqrt{2})i$	$(6+5\sqrt{2})-(2+2\sqrt{2})i$	$(6-5\sqrt{2})+(2\sqrt{2}-2)i$

4	-8	-12	$-20i$	$12+4i$	$-10\sqrt{2}-4\sqrt{2}i$	$12-4i$	$10\sqrt{2}-4\sqrt{2}i$

-8	-28	16	12	24	$-14\sqrt{2}(1+i)$	$8i$	$-6\sqrt{2}(1-i)$

16	-56	24	24	-32	0	8	0

6. Résultat:

(divide by 8)

2	-7	3	3	-4	0	1	0

Verification

In [1]:

```
from cmath import cos, sin, pi
from math import log2

def re_arange(a): # len(a) = 2^k, where k \in Z
    # there is a more efficient way to implement this using bit flipping
    if len(a) <= 2:
        return a
    b = [a[i] for i in range(0, len(a), 2)]
    c = [a[i+1] for i in range(0, len(a), 2)]
    return re_arange(b) + re_arange(c)

def omega(n, k):
    return complex(cos(2*pi*k/n), sin(2*pi*k/n))

def _FFT_Iter(arr, inv = False):
    n = len(arr)#end - start # length, shouldbe 2^k
    result = [None] * n
    if n == 0:
        return
    for k in range(n):
        k_prime = k if k < n//2 else k - n//2
        k_prime = -1 * k_prime if inv else k_prime
        if k < n//2:
            result[k] = arr[k] + omega(n, k_prime) * arr[k+n//2]
        else:
            result[k] = arr[k-n//2] - omega(n, k_prime) * arr[k]
    return result


def _FFT(arr, inv = False, verbose = True):
    n = len(arr)
    arr = re_arange(arr)
    if verbose:
        print("after re-order:{}\n".format(arr))
    n_iter = int(log2(n))
    for i in range(1, n_iter+1):  # log(n)
        sub_group_len = pow(2, i)
        n_sub_group = n // sub_group_len
        for j in range(n_sub_group):
            start, end = j*sub_group_len, (j+1)*sub_group_len
            sub_group = arr[start : end]
            arr[start : end] = _FFT_Iter(sub_group, inv)
        if verbose:
            print("Iter{}: {}\n".format(i, arr))
    return arr


def FFT(arr, verbose = True):
    return _FFT(arr, inv = False, verbose = verbose)


def IFFT(arr, verbose = True):
    a = _FFT(arr, inv = True, verbose = verbose)
    n = len(a)
    a_over_n = [a[i]/n for i in range(n)]
    return a_over_n
```

In [2]:

```
a = [1, -3, 0, 1, 0, 0, 0, 0]
b = [2, -1, 0, 1, 0, 0, 0, 0]
```

In [3]:
```
A, B = FFT(a), FFT(b)
```

```
after re-order:[1, 0, 0, 0, -3, 0, 1, 0]

Iter1: [(1+0j), (1+0j), 0j, 0j, (-3+0j), (-3+0j), (1+0j), (1+0j)]

Iter2: [(1+0j), (1+0j), (1+0j), (1+0j), (-2+0j), (-3+1j), (-4+0j), (-3-1j)]

Iter3: [(-1+0j), (-1.8284271247461903-1.414213562373095j), (0.9999999999999998-4j), (3.82842
712474619-1.4142135623730954j), (3+0j), (3.8284271247461903+1.414213562373095j), (1.00000000
00000002+4j), (-1.8284271247461898+1.4142135623730954j)]

after re-order:[2, 0, 0, 0, -1, 0, 1, 0]

Iter1: [(2+0j), (2+0j), 0j, 0j, (-1+0j), (-1+0j), (1+0j), (1+0j)]

Iter2: [(2+0j), (2+0j), (2+0j), (2+0j), 0j, (-0.9999999999999999+1j), (-2+0j), (-1-1j)]

Iter3: [(2+0j), (0.5857864376269051+2.220446049250313e-16j), (1.9999999999999998-2j), (3.414
213562373095-1.1102230246251565e-16j), (2+0j), (3.414213562373095-2.220446049250313e-16j),
(2+2j), (0.5857864376269051+1.1102230246251565e-16j)]
```

In [4]:
```
C = [A[i]*B[i] for i in range(len(A))]
print(C)
```

```
[(-2+0j), (-1.0710678118654753-0.8284271247461906j), (-6.000000000000001-9.999999999999998
j), (13.071067811865474-4.828427124746191j), (6+0j), (13.071067811865476+4.828427124746189
j), (-6+10j), (-1.0710678118654755+0.8284271247461903j)]
```

In [5]:
```
c = IFFT(C)
print(c)
```

```
after re-order:[(-2+0j), (6+0j), (-6.000000000000001-9.999999999999998j), (-6+10j), (-1.0710
678118654753-0.8284271247461906j), (13.071067811865476+4.828427124746189j), (13.071067811865
474-4.828427124746191j), (-1.0710678118654755+0.8284271247461903j)]

Iter1: [(4+0j), (-8+0j), (-12+1.7763568394002505e-15j), (-8.881784197001252e-16-20j), (12+3.
9999999999999982j), (-14.142135623730951-5.65685424949238j), (11.999999999999998-4j), (14.14
213562373095-5.6568542494923815j)]

Iter2: [(-8+1.7763568394002505e-15j), (-28-3.36468379447228e-16j), (16-1.7763568394002505e-1
5j), (12+3.36468379447228e-16j), (24-1.7763568394002505e-15j), (-19.79898987322333-19.798989
873223327j), (1.7763568394002505e-15+7.999999999999998j), (-8.485281374238571+8.485281374238
57j)]

Iter3: [(16+0j), (-56-3.36468379447228e-16j), (24-3.0628549591415595e-15j), (24+2.1128252188
474783e-15j), (-32+3.552713678800501e-15j), -3.36468379447228e-16j, (8.000000000000002-4.898
587196589412e-16j), -1.4398884599530224e-15j]

[(2+0j), (-7-4.20585474309035e-17j), (3-3.8285686989269494e-16j), (3+2.641031523559348e-16
j), (-4+4.440892098500626e-16j), -4.20585474309035e-17j, (1.0000000000000002-6.1232339957367
65e-17j), -1.799860574941278e-16j]
```

In [ ]:
```

```