HW Assignment - 9

Dhayalini Nagaraj
A20359686

1. What would happen in RECURSIVE-FFT if line 4 was changed to "$\omega_n = e^{2\pi q i/n}$"? That is find a simple relation between the output of RECURSIVE-FFT obtained with this change and the results obtained with the original procedure. (that is when $q=1$)

In RECURSIVE-FFT line 4 is

$$\omega_n = e^{2\pi i/n}$$

If line 4 is changed to $\omega_n = e^{2\pi q i/n}$

Let us subsititute $q=0$.

then $\omega_n = e^{2\pi(0)i/n}$

If $q=0$, then only one element in the objective vector can be obtained.

If $\gcd(q,n)=1$, then a permutation of the original vector can be got.

$\gcd(q,n)=1$, permutation of objective vector can be obtained.

If $\gcd(q, n) \neq 1$, all ~~roots~~ the elements in the original vector cannot be obtained.

(ie) We will not get all the elements in the original vector and only $\frac{n}{\gcd(q,n)}$ values can be got in the objective vector.

2. How many times does ITERATIVE-FFT compute twiddle factors in each stage? Rewrite ITERATIVE FFT to compute twiddle factors only 2^{s-1} times in stage s.

Line 6 - for $k = 0$ to $n-1$ by m

The loop in line 6 is executed n/m times.

Line-8 : for $j = 0$ to $m/2 - 1$

So, The loop in line 8 of ITERATIVE FFT is executed $m/2$ times.

$\therefore$ The twiddle factors are computed as $\frac{n}{m} \times \frac{m}{2} = n/2$ times.

To compute twiddle factors only 2^{s-1} times in stage s we should alter the loops in the ITERATIVE FFT

In the ITERATIVE FFT twiddle factors is calculated in the inner most loop.

In order to compute twiddle factors 2^{s-1} times in stage s, we can precompute a table of all needed twiddle factors instead of calculating it in the inner most loop.

ALGORITHM: ITERATIVE FFT MODIFIED (a)

```
BIT-REVERSE-COPY (a, A)
n = a.length        // n is a power of 2
for s = 1 to lg n
        m = 2^s
        ω_m = e^(2πi/m)
        T[0] = 1
        for i = 1 to m/2 - 1
            T[i] = T[i-1] · ω_m
        end for
        for K = 0 to n-1 by m
            for j = 0 to m/2 - 1
```

```
12.         t = T[j] · A[k+j + m/2]
13.         u = A[k+j]
14.         A[k+j] = u + t
15.         A[k+j+m/2] = u - t
16.       end for
17.     end for
18.   end for
19.   return A
```

In this algorithm lines 6-9 computes the twiddle factor for 2^{s-1} times in stage s.

3. a) In ITERATIVE-FFT, which twiddle factors(s) are computed with the most multiplications? How many multiplications is that?

The maximum number of iterations of the loop in line number 8 is $m/2$.

$$m/2 = 2^s/2 \leq 2^{\lg n}/2 = n/2$$

After $n/2$ multiplications the twiddle factor is $\omega_n^{n/2-1}$ because ω starts with $\omega_n^0 = 1$

3b) Assume that n is a power of 2, $n = 2^k$, and let ω_n be the principle n^{th} root of unity $\omega_n = e^{2\pi i/n}$

<u>To Prove:</u> $x_{r+1} = \sqrt{\frac{1+x_r}{2}}$

$$x_{r+1} = \cos\left(\frac{(2\pi/2^r)}{2}\right)$$

$$= \sqrt{\frac{1+\cos(2\pi/2^r)}{2}}$$

$$= \sqrt{\frac{1+x_r}{2}}.$$

<u>To Prove:</u> $y_{r+1} = \frac{y_r}{2x_{r+1}}$

$$y_{r+1} = \sin(2\pi/2^{r+1})$$

$$= \sin\left(\frac{(2\pi/2^r)}{2}\right)$$

$$= \frac{\sin(2\pi/2^r)}{2\cos(2\pi/2^{r+1})} = \frac{y_r}{2x_{r+1}}$$

3c) Show how using the binary representation of i, w_n^i can be computed by a product of at most k of the a_rs. Explain why this scheme uses $O(\log n)$ multiplications for each twiddle factor.

The binary representation of i with k bits is

$$i = \sum_{j=0}^{k-1} c_j 2^j, \quad c_j \in \{0,1\} \quad \text{and}$$

$$w_n^i = w_n^{\sum_{j=0}^{k-1} c_j 2^j}$$

$$= \prod_{j=0}^{k-1} w_n^{c_j 2^j} = \prod_{j=0}^{k-1} \alpha_k^{c_j 2^j}$$

$$= \prod_{j=0}^{k-1} \left(\alpha_k^{2^j}\right)^{c_j}$$

$$= \prod_{j=0}^{k-1} \alpha_{k-j}^{c_j}$$

From (a) subdivision we know

$i \leq n/2 - 1$ and its binary representation cannot have more than $\lfloor \log(n/2-1) \rfloor$ bits.

From the above analysis, the time for computing each twiddle is bounded by the number of bits.

$\lfloor \log(n/2-1) \rfloor$, which is $O(\log n)$

3d) Give modified version of algorithms RECURSIVE-FFT and ITERATIVE FFT that use the twiddle factors as precomputed in part (b) instead of computing them "on the fly".

For both the algorithms RECURSIVE FFT and ITERATIVE FFT we can precompute a table T of all needed twiddle factors by the methods in (b) and (c).

$T[k] = \omega_n^k$.

When the table is computed, we only need to look up the table for the needed twiddle factor.

ALGORITHM: RECURSIVE-FFT-MODIFIED (a)

1. n = a. length // n is a power of 2.
2. if n = = 1 then
3. return a
4. end if
5. $\omega_n = e^{2\pi i/n}$
6. $\omega = 1$
7. $a^{[0]} = (a_0, a_2, \ldots, a_{n-2})$
8. $a^{[0]} = (a_1, a_3, \ldots, a_{n-1})$
9. $y^{[0]}$ = RECURSIVE ($a^{[0]}$)
10. $y^{[1]}$ = RECURSIVE ($a^{[1]}$)
11. for k = 0 to n/2 − 1 do
12. $y_k = y_k^{[0]} + T[k]\, y_k^{[1]}$
13. $y_{k+n/2} = y_k^{[0]} - T[k]\, y_k^{[1]}$
14. end for
15. return A

ALGORITHM: ITERATIVE-FFT-MODIFIED (a)

1. BIT-REVERSE-COPY (a, A)
2. n = a.length // n is a power of 2.
3. for $s = 1$ to $\lg n$
4. $m = 2^s$
5. for $k = 0$ to $n-1$ by m
6. for $j = 0$ to $m/2 - 1$
7. $t = T[(n \cdot j)/m] \cdot A[k + j + m/2]$
8. $u = A[k+j]$
9. $A[k+j] = u + t$
10. $A[k+j+m/2] = u - t$
11. end for
12. end for
13. end for

References:

www.fftw.org
staff.ustc.edu.cn
stackoverflow.com/understanding-the fft-recursive-algorithms.
https://books.google.co.in/books?id=PPvWDFi3kxUC&pg.