1. In RECURSIVE-FFT(), if the value at line 4 of a algorithm is changed to $\omega_n = e^{2\pi q i/n}$ from $e^{2\pi i/n}$ then their can be two cases as follows:

(I) If the greatest common divisor between q and n turn out to be 1, that means q and n are not divisible by each other, then we will get original vector elements ($\vec{y}$) as their is no change in the relation. [$\omega = e^{2\pi i/n}$] {permutations of original vector}

original vector:

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & & 1 \\ 1 & \omega_n & \omega^2_n & & \omega_n^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & & \omega_n^{(n-1)(n-1)} \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

(II) If q and n are divisors of each other then their GCD will not be equal to 1 and the original vector can loose some permutations of original vector.

2. In ITERATIVE-FFT() algorithm k loop will be executed n/m times and loop j will be executed $m/2$ times.

$\therefore$ Twiddle factor for each iteration will be computed ~~as~~

$$\frac{n}{m} * \frac{m}{2} = \frac{n}{2^s} * \frac{2^s}{2} = \frac{n}{2^s} * 2^{s-1} = \boxed{\frac{n}{2}}$$

where $m = 2^s$

2. We can avoid computations in the innermost loops if we compute all powers < m/2 of ω_m before k loop (line 6).

Algorithm : ITERATIVE-FFT-UPDATED (a)

```
1.  BIT-REVERSE-COPY(a, A)
2.  n = a.length                  // n is power of 2
3.  for s = 1 to log n
4.      m = 2^s
5.      ωm = e^(2πi/m)
6.      F[0] = 1                  // initialize twiddle factor &
                                     set index 0 value to 1
                                     for table F
7.      for x = 1 to m/2 - 1 do
8.          F[x] = F[x-1] · ωm    // ω = ω·ωm
9.      end for
10.     for k = 0 to n-1 by m do
11.         for j = 0 to m/2 - 1 do
12.             t = F[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
```

- Twiddle factor computed = 2^{s-1} times {from line 6 to 10}

3. (a) In ITERATIVE-FFT(a) algorithm, the outer 'k' loop at line 6 cannot be considered for most multiplications required for twiddle factor ~~computation~~ computation as it sets $\omega = 1$ each time a stage is completed.

- The j loop at line 8 used to computed internal groups ~~can~~ be considered for twiddle factor computations. (max multiplications)
- The max value computed by loop 'j' would be $m/2$
- As given $m = 2^s$ & max limit for s loop $= \log n$

$$\therefore \frac{m}{2} = \frac{2^s}{2} \leq \frac{2^{\log n}}{2} = \boxed{\frac{n}{2}}$$

- $\omega_n^0 = 1$. Therefore, twiddle factor is $\boxed{\omega_n^{n/2-1}}$

(b) Given:

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

$$a_r = e^{2\pi i/2^r}$$

$$e^{i\theta} = \cos\theta + i\sin\theta$$

$$\therefore e^{2\pi i/2^r} = \cos\left(\frac{2\pi}{2^r}\right) + i\sin\left(\frac{2\pi}{2^r}\right)$$

$$a_r = x_r + iy_r$$

$$\therefore a_{r+1} = x_{r+1} + iy_{r+1} \leftarrow \text{to be proved}$$

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

~~3. (a)~~

3 . (b) $= \cos\left(\frac{2\pi/2^r}{2}\right) + i\sin\left(\frac{2\pi/2^r}{2}\right)$

As we know; $\frac{\cos\theta}{2} = \sqrt{\frac{1+\cos\theta}{2}}$ —— (1)

$\sin 2\theta = 2\cos\theta \cdot \sin\theta$

$\therefore \sin\theta = \frac{\sin 2\theta}{2\cos\theta}$ —— (2)

From (1) & (2);

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

$$= \sqrt{\frac{1+\cos(2\pi/2^r)}{2}} + i\,\frac{y_r}{2 \cdot x_{r+1}}$$

$$\therefore a_{r+1} = x_{r+1} + i\,y_{r+1}$$

(c) Binary Representation $\{0, 1\}$ of i having k bits can be represented as;

$$i = \sum_{j=0}^{n-1} b_j 2^j$$

$\{$ Polynomial in variable x over an algebric field is represented by a function $A(x) = \sum_{j=0}^{n-1} a_j x^j$ $\}$

where,

b_j belongs to $\{0, 1\}$ binary set.

3. (c) $\therefore \omega_n^i = \omega^{\sum_{j=0}^{n-1} c_j 2^j}$

$\therefore$ product of above sumation/sum;

$$= \prod_{j=0}^{n-1} \omega_n^{c_j 2^j}$$

$$= \prod_{j=0}^{n-1} \alpha_n^{c_j 2^j} \qquad \left\{ \begin{array}{l} \omega_n = e^{2\pi i/n} \qquad \text{here } [r \cong n] \\ \alpha_r = e^{2\pi i/2r} \\ \therefore \omega_n = \alpha_r \quad \text{from (3 (b))} \end{array} \right\}$$

$$= \prod_{j=0}^{n-1} (\alpha_n^{2^j})^{c_j}$$

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

From (a) of question 3, we know max limit of stage $s = \log_n$ & $i \leq n/2 - 1$

$\therefore$ From above analysis, twiddle factor can be computed by number of bits.

i.e. $\log(n/2 - 1) \cong O(\log n)$

3. (d) We know, $F[k] = \omega^k_n$ from (b) & (c)

- Pre computation of table F of required twidle Factors. for both RECURSIVE-FFT and ITERATIVE-FFT as mentioned in (b) part

(1) Algorithm : RECURSIVE-FFT-UPDATED (a)

```
n = a.length
if n == 1 → return a
end if
ωn = e^(2πi/n)
ω = 1
a[0] = (a0, a2 ..., an-2)
a[1] = (a1, a3 ...., an-1)
y[0] = RECURSIVE-FFT-UPDATED (a[0])
y[1] = RECURSIVE-FFT-UPDATED (a[1])
For k = 0 to n/2 - 1
    yk = yk^[0] + F[k] yk^[1]
    yk+n/2 = yk^[0] - T[k] yk^[1]
end for
Return y
```

(2) Algorithm: ITERATIVE-FFT-UPDATED (a,A)

```
Bit-Reverse-Copy
n = a.length
For s = 1 to log n
    m = 2^s
    For k = 0 to n-1 by m
        For j = 0 to m/2 - 1
            t = F[(n*j)/m] · A[k+j+m/2]
            u = A[k+j]
```

3. (d)

```
9.      A[k+j] = u+t
10.     A[k+j+m/2] = u-t
11.    end for
12.    end for
13. end for
15. return A
```