DFT: $\mathbb{R}[x]/x^N+1 \longrightarrow \mathbb{C}^{\frac{N}{2}}$

$\Downarrow$ $\qquad$ $\Downarrow$

$m(x)$ $\qquad$ $Z = [m(\zeta), m(\zeta^5), \cdots, m(\zeta^{5^{\frac{N}{2}}})]$

$= m[0] + m[1]x + \cdots + m[N-1]x^{N-1}$ $\qquad$ $\zeta = e^{\frac{2\pi i}{2N}}$

DFT: $\begin{bmatrix} Z \\ \bar{Z} \end{bmatrix} = \begin{bmatrix} U_0 & iU_0 \\ \bar{U}_0 & \bar{iU}_0 \end{bmatrix}_{[N/2 \; N/2]} \begin{bmatrix} m \end{bmatrix}$, $Z = \underline{U_0 m^{1st}} + \underline{iU_0 m^{2nd}}$

$U_0 \bar{U}_0^T = \frac{N}{2} \cdot I$, $\qquad \bar{U}_0^T Z = \frac{N}{2} m^{1st} + i\frac{N}{2} m^{2nd}$

IDFT: $\qquad m^{1st} = \frac{2}{N} \cdot \text{realpart}(\bar{U}_0^T Z)$

$m^{2nd} = \frac{2}{N} \cdot \text{imagpart}(\bar{U}_0^T Z)$

$U_0 R = E_2^{(N/2)} \; E_4^{(N/2)} \cdots E_{N/2}^{(N/2)}$

$E_R^{(n)} = \begin{bmatrix} \begin{bmatrix} I_{n/R} & W_{n/R} \\ I_{n/R} & -W_{n/R} \end{bmatrix} & & \\ & \ddots & \\ & & \ddots \end{bmatrix}$

: Sparse with 3 diagonal vectors.

$W_n = \text{diag}(\omega_{4n}^{5^i})_{0 \le i < n}$

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

R: bit-reversal permutation matrix

$$R \begin{bmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \end{bmatrix} = \begin{bmatrix} Z_0 \\ Z_2 \\ Z_1 \\ Z_3 \end{bmatrix}$$

```
#define PI 3.14159265358979323846
```

$$Z = U_0 (m^{1st} + i m^{2nd})$$

$$U_0[i][j] = \exp\left(\frac{2\pi i}{2N} \times (5^i \cdot j)\right)$$

```
template<int N>
void dft(const double m[N], double zr[N/2],
                            double zi[N/2]){
    for(int i=0, poweri=1; i<N/2; i++, poweri=(5*poweri)%(2*N)){
        zr[i]=0; zi[i]=0;
        for(int j=0, powerij=0; j<N/2; j++, powerij=(powerij+poweri)%(2*N)){
            double c=cos(PI/N*powerij);
            double s=sin(PI/N*powerij);
            zr[i]+= c*m[j] - s*m[j+N/2];
            zi[i]+= c*m[j+N/2] + s*m[j];
        }
    }
}
```

$$m^{1st} + i m^{2nd} = \frac{2}{N} \cdot \bar{U}_0^T \cdot Z, \quad \bar{U}_0^T[i][j] = \exp\left(\frac{2\pi i}{2N} \times (-5^j \cdot i)\right)$$

```
template<int N>
void idft(const double zr[N/2],
          const double zi[N/2], double m[N]){
    for(int i=0; i<N/2; i++){
        m[i]=0; m[i+N/2]=0;
        for(int j=0, fiveji=i; j<N/2; j++, fiveji=(fiveji*5)%(2*N)){
            double c=cos(PI/N*powerij);
            double s=sin(PI/N*powerij);
            m[i] += c*zr[j] + s*zi[j];
            m[i+N/2]+= -s*zr[j] + c*zi[j];
        }
        m[i] *= 2./N;
        m[i+N/2] *= 2./N;
    }
}
```

```
template <int N>
void bitReverse( const double a[N], double b[N]){
    int logN=0;
    while(N>(1<<logN))
        logN++;
    for( int i=0; i<N; i++) {
        int j=0;
        for(int k=0; k<logN; k++)
            j+=((i>>k) & 1)<<(logN-1-k);
        b[j]=a[i];
    }
}
```

logN 개수

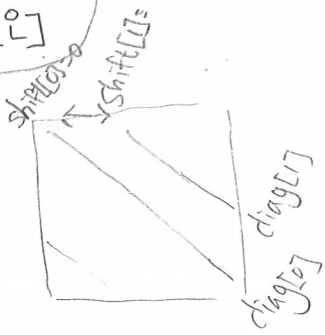
N=8, i=110

logN=3

j= 0<<2 + 1<<1 + 1<<0

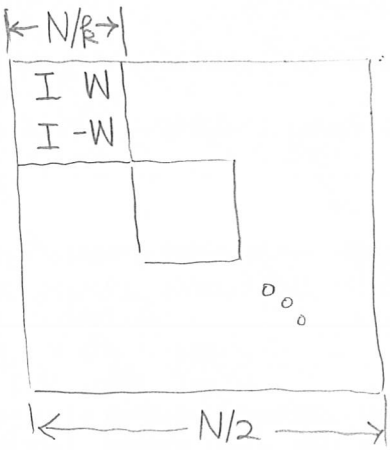
```
template<int N, int NumDigs>
struct SparseComplexMatrix{
    int shift[NumDigs];
    double diagr[ // ][N];
    double diagi[ // ][N];
    SparseComplexMatrix(){
        for(int k=0; k<NumDigs; k++){ shift[k]=0;
        for(int i=0; i<N; i++){ diagr[k][i]=0;
                                diagi[k][i]=0;}}
    }
    void applyA( const double xr[N]
                 const double xi[N], double Axr[N],
                                     double Axi[N]){
        for(int i=0; i<N; i++){
            Axr[i]=0; Axi[i]=0;
            for(int k=0; k<NumDigs; k++){
                int j=(i+shift[k])%N;
                double aijr= diagr[k][i];
                double aiji= diagi[k][i];
                Axr[i]+= aijr*xr[j] - aiji*xi[j];
                Axi[i]+= aijr*xi[j] + aiji*xr[j];
            }
        }
    }
};
```

A[i][i + shift[k]] = diag[k][i]



$$U_0R = E_2^{N/2} E_4^{N/2} \cdots E_{N/2}^{N/2}$$

$$E_R^{N/2} =$$



I: $N/2R \times N/2R$

$W = \mathrm{diag}(\zeta^{5^i})_{0 \le i < \frac{N}{2R}}$

$\zeta = \exp\left(\frac{2\pi i}{2N} \cdot \frac{R}{2}\right)$

```
template < int LOGN >
void splitUOR ( SparseComplexMatrix < 1<<(LOGN-1), 3> E[LOGN-1] ) {
    int N = 1<<LOGN;
    for(int R=2, r=0; R<=N/2; R*=2, r++){
        int M=N/2/R;  E[r].shift[0]=0; E[r].shift[1]=M; E[r].shift[2]
                                                        =N/2-M;
```

W block: M×M

```
        double Wr[1<<(LOGN-2)];
        double Wi[1<<(LOGN-2)];
        for(int i=0, fiveiR=R/2; i<M; i++, fiveiR=(fiveiR*5)%(2*N)){
            Wr[i] = cos(PI/N*fiveiR);
            Wi[i] = sin(PI/N*fiveiR);
        }
```

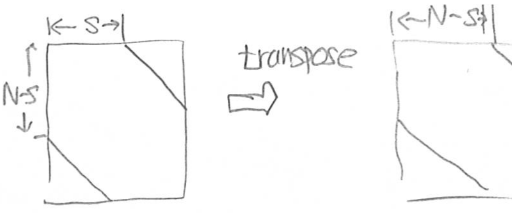
block $\frac{R}{2}$-TH

I	W
I	-W

```
        for(int b=0; b<R/2; b++){
            for(int i=0; i<M; i++){
                E[r].diagr[0][2*M*b+ i] = 1;
                E[r].diagr[2][2*M*b+ M+i] = 1;
                E[r].diagr[1][2*M*b + i] = Wr[i];
                E[r].diagi[1][2*M*b + i] = Wi[i];
                E[r].diagr[0][2*M*b+ M+i] = -Wr[i];
                E[r].diagi[0][   "    ] = -Wi[i];
            }
        }
    }
}
```

#5

```
template <int N, int NumDigs>
void SparseComplexMatrix::transpose(){
    for(int k=0; k<NumDigs; k++){
        int s= shift[k];
        shift[k]=(N-s)%N;
        double tempr[N], tempi[N];
        for(int i=0; i<N; i++){ tempr[i]= diagr[k][i];
                                tempi[i]= diagi[k][i]; }
        for(int i=0; i<N; i++){ diagr[k][i]= tempr[(i-s)%N];   // i+N
                                diagi[k][i]=tempi[    "    ]; }
    }
}
```




```
template <int N, int NumDigs>
void SparseComplexMatrix::conjugate(){
    for(int k=0; k<NumDigs; k++)
    for(int i=0; i<N; i++) diagi[k][i]= -diagi[k][i];
}
```

```
template<int N, int LOGN>
void fft( const double m[N], double zr[N/2], double zi[N/2]){
    SparseComplexMatrix <N/2, 3> E[LOGN-1];  splitUONR<LOGN>(E);
    bitReverse <N/2>(m    , zr );
    bitReverse <N/2>(m+N/2, zi );
    double tempr[N/2], tempi[N/2];
    for(int k=LOGN-2; k>=0; k--){
        for(int i=0; i<N/2; i++){ tempr[i]=zr[i];  tempi[i]=zi[i]; }
        E[k].apply(tempr, tempi, zr, zi);
    }
}
```

$U_0 = E[0] \cdots E[LOGN-2] \cdot R$

```
template<int N, int LOGN >
void ifft(const double zr[N/2], const double zi[N/2], double m[N]){
    SparseComplexMatrix <N/2,3> E[LOGN-1];  splitUONR <LOGN> (E);
    double tempr[N/2], tempi [N/2];  for(int i=0; i< N/2; i++){ tempr[i]=zr[i]; tempi[i]=zi[i]; }
    for(int k=0; k< LOGN-1; k++){
        E[k].transpose();  E[k].conjugate();
        E[k].apply(tempr, tempi, m, m+N/2);
        for(int i=0; i<N/2; i++){ tempr[i]=m[i];  tempi[i]= m[N/2+i]; }
    }
    bitReverse<N/2>(tempr, m);  bitReverse< N/2>(tempi, m+N/2);
    for(int i=0; i<N; i++)  m[i] *= 2./N;
}
```

$\bar{U}_0^T = R\ \bar{E}^T[LOGN-2] \cdots \bar{E}^T[0]$