

# FFT1 C code

September 8, 2021 9:50 PM

```

#define LONG long long int
int mulmod( int a, int b, int p ) {
    int t = (LONG) a * b % p;
    return t;
}
int addmod( int a, int b, int p ) {
    int t = a+b;
    t += (t>>31) & p; // if( t<0 ) t+= p;
    return t;
}
int submod( int a, int b, int p ) {
    int t = a-b;
    t += (t>>31) & p; // if( t<0 ) t+= p;
    return t;
}

```

$\mathbb{Z}_p$

$a \cdot b \pmod p$

$a+b \pmod p$

$a-b \pmod p$

input. ↘ ↙ output

inplace implementation.

```

void FFT1( int *A, // [a0,a1,...,a_{n/2},0,...,0] of size n
           int n, // n = 2^k
           int *W, // [ powers of w, w^2, w^4, w^8, ... ]
           int p, // prime < 2^31
           int *T ) // [ scratch array of size n ]
{
    int i,n2,t;
    if( n==1 ) return;
    n2 = n/2;

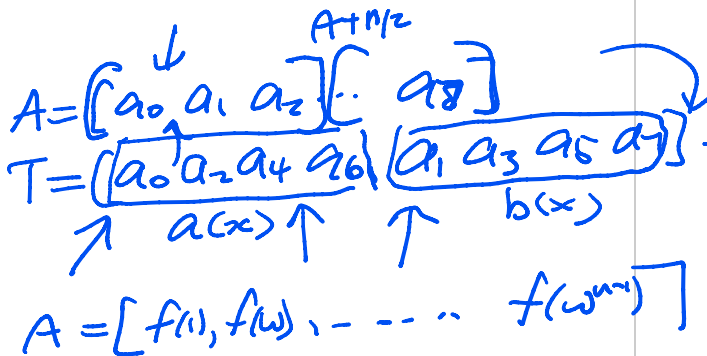
    for( i=0; i<n2; i++ ) {
        T[ i ] = A[2*i];
        T[ n2+i ] = A[2*i+1];
    }

    FFT1( T, n2, W+n2, p, A );
    FFT1( T+n2, n2, W+n2, p, A+n2 );

    for( i=0; i<n2; i++ ) {
        t = mulmod(W[i],T[n2+i],p);
        A[ i ] = addmod(T[i],t,p);
        A[ n2+i ] = submod(T[i],t,p);
    }
    return;
}

```

$n=8$



$W = [1, \omega, \omega^2, \dots, \omega^{n/2-1} \quad \downarrow \omega^{n/2} \quad [1, \omega^2, \omega^4, \dots, \omega^{n/2-2}, 1, \omega^4, \omega^8, \dots, \omega^{n/2-4}, \dots, 1, 0]]$