DSIP

Fast Fourier Transform

Code:

```
// Algorithm Used
// X0,...,N−1 ← ditfft2(x, N, s):             DFT of (x0, xs, x2s, ..., x(N-1)s):
//    if N = 1 then
//        X0 ← x0                              trivial size-1 DFT base case
//    else
//        X0,...,N/2−1 ← ditfft2(x, N/2, 2s)            DFT of (x0, x2s, x4s, ...)
//        XN/2,...,N−1 ← ditfft2(x+s, N/2, 2s)          DFT of (xs, xs+2s, xs+4s, ...)
//        for k = 0 to N/2−1                            combine DFTs of two halves into full DFT:
//            t ← Xk
//            Xk ← t + exp(−2πi k/N) Xk+N/2
//            Xk+N/2 ← t − exp(−2πi k/N) Xk+N/2
//        endfor
//    endif

#include <stdio.h>
#include <complex.h>
#include <math.h>

double PI;
typedef double complex cplx;

void fft(cplx x[], int N, int s, cplx X[]) {
	int k;
	cplx t;
	if (N == 1)
		X[0] = x[0];
	else {
		fft(x, N/2, 2*s, X);
		fft(x+s, N/2, 2*s, X+(N/2));
		for (k = 0; k < N/2; k++) {
			t = X[k];
			X[k] = t + cexp(-2*PI*I*k / N) * X[k + N/2];
			X[k + N/2] = t - cexp(-2*PI*I*k / N) * X[k + N/2];
		}
	}
}

void show(const char * s, cplx buf[]) {
	int i;
	printf("%s", s);
	for (i = 0; i < 8; i++)
		if (!cimag(buf[i]))
			printf("%g ", creal(buf[i]));
		else
			printf("(%g, %g) ", creal(buf[i]), cimag(buf[i]));
}

int main() {
	PI = atan2(1, 1) * 4;
	cplx x[] = {1, 2, 3, 4, 4, 3, 2, 1};
	cplx X[8];
	show("Data: ", x);
	fft(x, 8, 1, X);
	show("\nResult: ", X);
	printf("\n");
	return 0;
}
```

Output:

```
parth@parth-Inspiron-5565:~/College/Semester_7/DSIP/Fast_Fourier_Transform$ ./fft
Data: 1 2 3 4 4 3 2 1
Result: 20 (-5.82843, -2.41421) 0 (-0.171573, -0.414214) 0 (-0.171573, 0.414214) 0 (-5.82843, 2.41421)
```