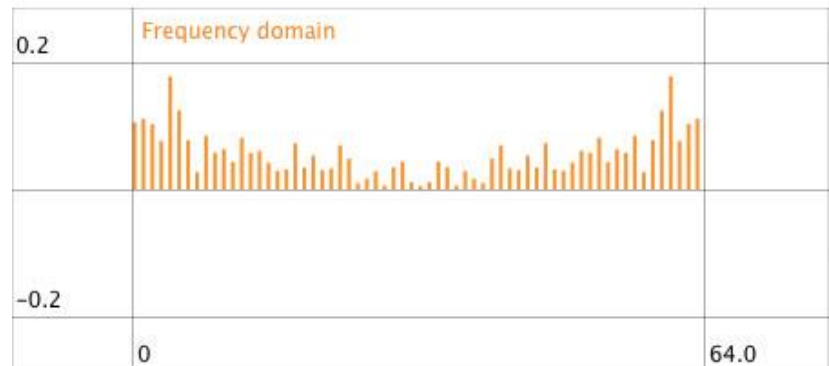


Discrete Fourier Transform(DFT)

- DFT applies Fourier transform to digitized data.
- The signal is transformed to time domain to frequency domain.
- IDFT is Inverse DFT.



DFT(1)

- Discrete Fourier Transform(DFT)
 - Time domain $\rightarrow$ Frequency domain

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}$$

- Inverse DFT (iDFT)
 - Frequency domain $\rightarrow$ Time domain

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn}$$

DFT(2)

- Apply Euler's formula

$$e^{-j\frac{2\pi}{N}kn} = \cos\left(\frac{2\pi}{N}kn\right) - j \sin\left(\frac{2\pi}{N}kn\right)$$

- DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) \left(\cos\left(\frac{2\pi}{N}kn\right) - j \sin\left(\frac{2\pi}{N}kn\right) \right)$$

- iDFT

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \left(\cos\left(\frac{2\pi}{N}kn\right) + j \sin\left(\frac{2\pi}{N}kn\right) \right)$$

DFT(3)

- Separation of Real Part and Imaginary Part

$$\left. \begin{aligned} X_{re}(k) &= \sum_{n=0}^{N-1} \left(x_{re}(n) \cos\left(\frac{2\pi}{N}kn\right) + x_{im}(n) \sin\left(\frac{2\pi}{N}kn\right) \right) \\ X_{im}(k) &= j \sum_{n=0}^{N-1} \left(-x_{re}(n) \sin\left(\frac{2\pi}{N}kn\right) + x_{im}(n) \cos\left(\frac{2\pi}{N}kn\right) \right) \end{aligned} \right\} \text{DFT}$$
$$\left. \begin{aligned} x_{re}(n) &= \frac{1}{N} \sum_{k=0}^{N-1} \left(X_{re}(k) \cos\left(\frac{2\pi}{N}kn\right) - X_{im}(k) \sin\left(\frac{2\pi}{N}kn\right) \right) \\ x_{im}(n) &= j \frac{1}{N} \sum_{k=0}^{N-1} \left(X_{re}(k) \sin\left(\frac{2\pi}{N}kn\right) + X_{im}(k) \cos\left(\frac{2\pi}{N}kn\right) \right) \end{aligned} \right\} \text{iDFT}$$

DFT(4)

- Separation of real part and imaginary part

$$\begin{aligned} X_{re}(k) &= \sum_{n=0}^{N-1} \left(x_{re}(n) \cos\left(\frac{2\pi}{N}kn\right) + x_{im}(n) \sin\left(\frac{2\pi}{N}kn\right) \right) \\ X_{im}(k) &= j \sum_{n=0}^{N-1} \left(-x_{re}(n) \sin\left(\frac{2\pi}{N}kn\right) + x_{im}(n) \cos\left(\frac{2\pi}{N}kn\right) \right) \end{aligned} \quad \left. \vphantom{\sum_{n=0}^{N-1}} \right\} \text{DFT}$$
$$\begin{aligned} x_{re}(n) &= \frac{1}{N} \sum_{k=0}^{N-1} \left(X_{re}(k) \cos\left(\frac{2\pi}{N}kn\right) - X_{im}(k) \sin\left(\frac{2\pi}{N}kn\right) \right) \\ x_{im}(n) &= j \frac{1}{N} \sum_{k=0}^{N-1} \left(X_{re}(k) \sin\left(\frac{2\pi}{N}kn\right) + X_{im}(k) \cos\left(\frac{2\pi}{N}kn\right) \right) \end{aligned} \quad \left. \vphantom{\sum_{k=0}^{N-1}} \right\} \text{iDFT}$$

DFT(5)

```
for(i = 0; i < num; i++){
    for(j = 0; j < num; j++){
        temp_re[i] += re[j]*cos(2*PI*i*j/num + flag*im[j]*sin(2*PI*i*j/num);
        temp_im[i] += -flag*re[j]*sin(2*PI*i*j/num) + im[j]*cos(2*PI*i*j/num);
        // if DFT, flag = 1, else if iDFT, flag == -1
    }
    if(iDFT){
        temp_re[i] /= num;
        temp_im[i] /= num;
    }
}
for(i = 0; i < num; i++){
    re[i] = temp_re[i];
    im[i] = temp_im[i];
}
```

Contest Program

- `$ ./dft -n 8 -s 3`
- the number of elements : 1024
- the name of dataset : data_3.txt

Options...

- n [1-8] : the number of elements
- s [1-3] : select data set

- Elapsed time on CPU (DFT) : 167.144287 [msec]
- Elapsed time on CPU (IDFT) : 167.164734 [msec]

- Elapsed time on GPU (DFT) : 1.763616 [msec]
- Elapsed time on GPU (IDFT) : 1.693888 [msec]

Elapsed time & acceleration ratio

- 94.773628 times faster on dft!!
- 94.785225 times faster on idft!!

- Degree of similarity = 0.998622
- correct !!
- When this result is close to 1, the calculation is correcter

Verify the results...
Compute the similarity
similarity is 1 ~ -1

Contest

- Minimum Requirement
 - Accelerating DFT & IDFT by using GPU
 - Modify only `gpu_calc.cu`
 - Not have to execute all parts on GPU
 - Initialization and verification supported by toolkit
- Advanced
 - Optimization to achieve higher performance

Toolkit(Code)

- `gpu_calc.cu`
 - DFT & iDFT program for GPU
 - not implemented (please modify this file)
- `gpu_calc.cpp`
 - DFT & iDFT program for CPU
 - Refer to modify `gpu_calc.cu`
- `main.c`
 - Initialize & call functions

How to use toolkit

- make
 - Compile
- ./dft
 - run your program with default parameter
- ./dft -n [num] -s [num]
 - run your program with selected parameter
- ./dft -h
 - Help

Deadline

- 8/3 24:00
- Leave your design on your account
- Mail to kaneda@am.ics.keio.ac.jp with a simple report including:
 - Result of speed up
 - How did you try to improve the performance
- No submission, no unit.
- The contest results will be announced asap after the deadline.

Tips for parallel processing

- DFT is consisting of double loops.
- You should parallelize the outer loop first.
- For parallelize the inner loop, you should use reduction calculation well.