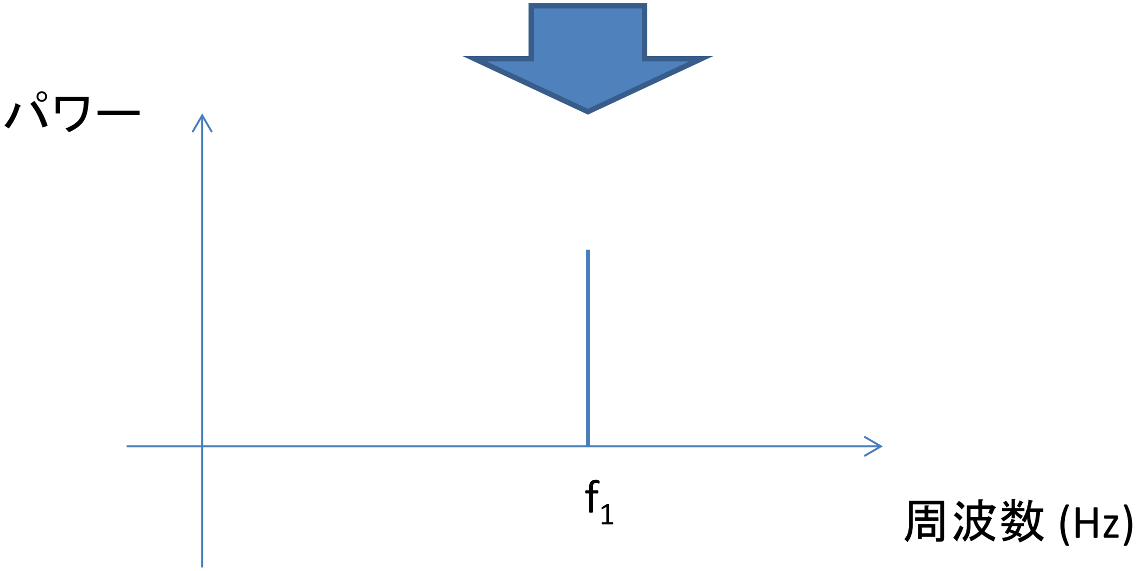
# フーリエ変換

# フーリエ変換

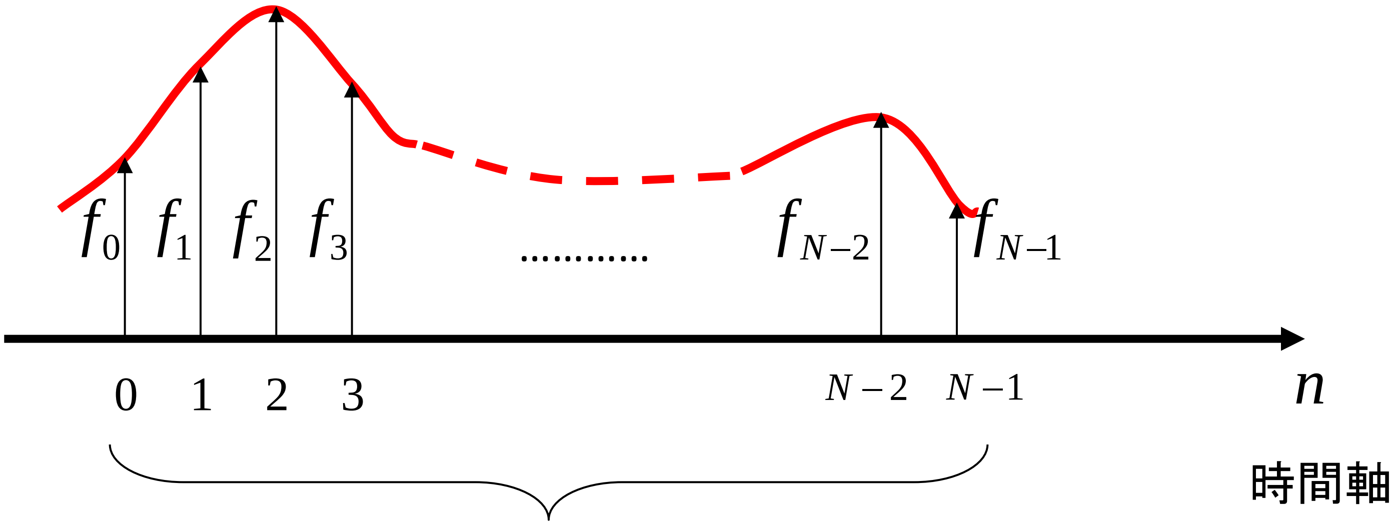
- ある信号の周波数成分を求める方法

$$y(t) = \sin(2\pi f_1 t)$$



フーリエ変換によって求まるパワースペクトル

# １次元離散的フーリエ変換



定義：

$$F_k = \sum_{n=0}^{N-1} f_n W_N^{nk}$$

$$W_N = e^{-j\frac{2\pi}{N}}$$

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

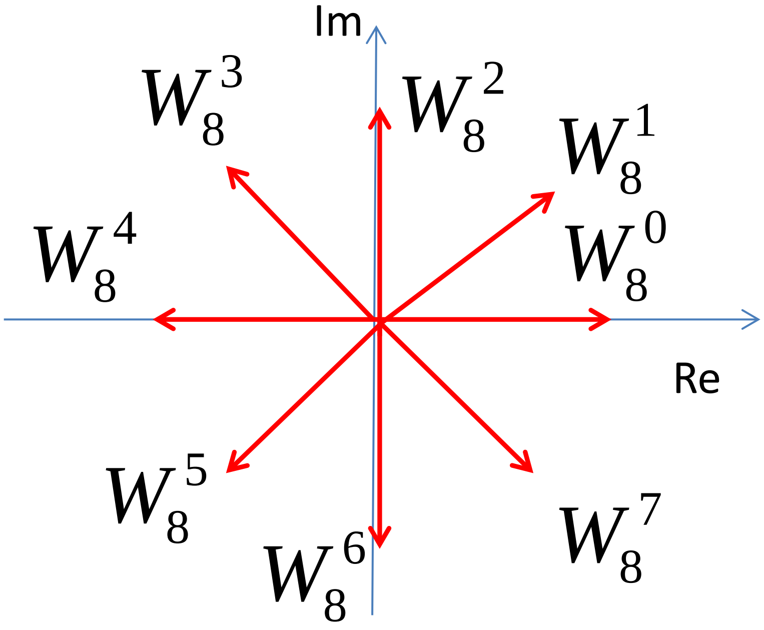
$$
\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix}
=
\begin{bmatrix}
W_8^{0} & W_8^{0} & W_8^{0} & W_8^{0} & W_8^{0} & W_8^{0} & W_8^{0} & W_8^{0} \\
W_8^{0} & W_8^{1} & W_8^{2} & W_8^{3} & W_8^{4} & W_8^{5} & W_8^{6} & W_8^{7} \\
W_8^{0} & W_8^{2} & W_8^{4} & W_8^{6} & W_8^{8} & W_8^{10} & W_8^{12} & W_8^{14} \\
W_8^{0} & W_8^{3} & W_8^{6} & W_8^{9} & W_8^{12} & W_8^{15} & W_8^{18} & W_8^{21} \\
W_8^{0} & W_8^{4} & W_8^{8} & W_8^{12} & W_8^{16} & W_8^{20} & W_8^{24} & W_8^{28} \\
W_8^{0} & W_8^{5} & W_8^{10} & W_8^{15} & W_8^{20} & W_8^{25} & W_8^{30} & W_8^{35} \\
W_8^{0} & W_8^{6} & W_8^{12} & W_8^{18} & W_8^{24} & W_8^{30} & W_8^{36} & W_8^{42} \\
W_8^{0} & W_8^{7} & W_8^{14} & W_8^{21} & W_8^{28} & W_8^{35} & W_8^{42} & W_8^{49}
\end{bmatrix}
\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}
$$

８×８＝64回の乗算が必要

一般には高速フーリエ変換が利用される

# 回転子 W

例えば $W_8^{49} = W_8^{9} = W_8^{2+7}$ と表記できる



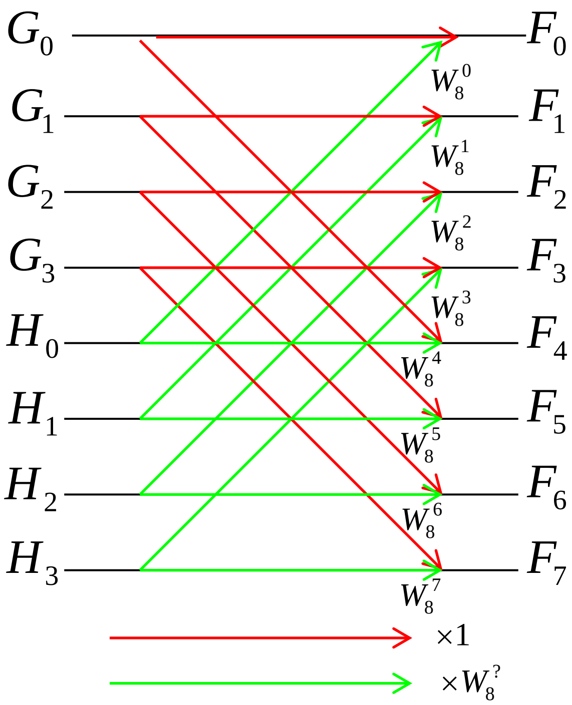
# 奇数と偶数に並び替え

$$\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix} = \left[\begin{array}{cccccccc} W_8^{0} & W_8^{0} & W_8^{0} & W_8^{0} & W_8^{0+0} & W_8^{0+0} & W_8^{0+0} & W_8^{0+0} \\ W_8^{0} & W_8^{2} & W_8^{4} & W_8^{6} & W_8^{0+1} & W_8^{2+1} & W_8^{4+1} & W_8^{6+1} \\ W_8^{0} & W_8^{4} & W_8^{0} & W_8^{4} & W_8^{0+2} & W_8^{4+2} & W_8^{0+2} & W_8^{4+2} \\ W_8^{0} & W_8^{6} & W_8^{4} & W_8^{2} & W_8^{0+3} & W_8^{6+3} & W_8^{4+3} & W_8^{2+3} \\ \hdashline W_8^{0} & W_8^{0} & W_8^{0} & W_8^{0} & W_8^{0+4} & W_8^{0+4} & W_8^{0+4} & W_8^{0+4} \\ W_8^{0} & W_8^{2} & W_8^{4} & W_8^{6} & W_8^{0+5} & W_8^{2+5} & W_8^{4+5} & W_8^{6+5} \\ W_8^{0} & W_8^{4} & W_8^{0} & W_8^{4} & W_8^{0+6} & W_8^{4+6} & W_8^{0+6} & W_8^{4+6} \\ W_8^{0} & W_8^{6} & W_8^{4} & W_8^{2} & W_8^{0+7} & W_8^{6+7} & W_8^{4+7} & W_8^{2+7} \end{array}\right] \begin{bmatrix} f_0 \\ f_2 \\ f_4 \\ f_6 \\ f_1 \\ f_3 \\ f_5 \\ f_7 \end{bmatrix}$$

時間間引き　４点のDFT

$$\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix} = \begin{bmatrix} \left(f_0W_8^0 + f_2W_8^0 + f_4W_8^0 + f_6W_8^0\right) + \left(f_1W_8^0 + f_3W_8^0 + f_5W_8^0 + f_7W_8^0\right)W_8^0 \\ \left(f_0W_8^0 + f_2W_8^2 + f_4W_8^4 + f_6W_8^6\right) + \left(f_1W_8^0 + f_3W_8^2 + f_5W_8^4 + f_7W_8^6\right)W_8^1 \\ \left(f_0W_8^0 + f_2W_8^4 + f_4W_8^0 + f_6W_8^4\right) + \left(f_1W_8^0 + f_3W_8^4 + f_5W_8^0 + f_7W_8^4\right)W_8^2 \\ \left(f_0W_8^0 + f_2W_8^6 + f_4W_8^4 + f_6W_8^2\right) + \left(f_1W_8^0 + f_3W_8^6 + f_5W_8^4 + f_7W_8^2\right)W_8^3 \\ \left(f_0W_8^0 + f_2W_8^0 + f_4W_8^0 + f_6W_8^0\right) + \left(f_1W_8^0 + f_3W_8^0 + f_5W_8^0 + f_7W_8^0\right)W_8^4 \\ \left(f_0W_8^0 + f_2W_8^2 + f_4W_8^4 + f_6W_8^6\right) + \left(f_1W_8^0 + f_3W_8^2 + f_5W_8^4 + f_7W_8^6\right)W_8^5 \\ \left(f_0W_8^0 + f_2W_8^6 + f_4W_8^4 + f_6W_8^2\right) + \left(f_1W_8^0 + f_3W_8^6 + f_5W_8^4 + f_7W_8^2\right)W_8^6 \\ \left(f_0W_8^0 + f_2W_8^6 + f_4W_8^4 + f_6W_8^2\right) + \left(f_1W_8^0 + f_3W_8^6 + f_5W_8^4 + f_7W_8^2\right)W_8^7 \end{bmatrix}$$

$$\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix} = \begin{bmatrix} G_0 + H_0W_8^0 \\ G_1 + H_1W_8^1 \\ G_2 + H_2W_8^2 \\ G_3 + H_3W_8^3 \\ G_0 + H_0W_8^4 \\ G_1 + H_1W_8^5 \\ G_2 + H_2W_8^6 \\ G_3 + H_3W_8^7 \end{bmatrix}$$



# 奇数行と偶数行に分ける

$$\begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix} = \begin{bmatrix} W_8^0 & W_8^0 & W_8^0 & W_8^0 \\ W_8^0 & W_8^2 & W_8^4 & W_8^6 \\ W_8^0 & W_8^4 & W_8^0 & W_8^4 \\ W_8^0 & W_8^6 & W_8^4 & W_8^2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_2 \\ f_4 \\ f_6 \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^0 & W_4^2 \\ W_4^0 & W_4^3 & W_4^2 & W_4^1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_2 \\ f_4 \\ f_6 \end{bmatrix}$$

$$\begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^2 & W_4^1 & W_4^3 \\ W_4^0 & W_4^0 & W_4^2 & W_4^2 \\ W_4^0 & W_4^2 & W_4^3 & W_4^1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_4 \\ f_2 \\ f_6 \end{bmatrix} = \left[ \begin{array}{cccc} W_4^0 & W_4^0 & W_4^{0+0} & W_4^{0+0} \\ W_4^0 & W_4^2 & W_4^{0+1} & W_4^{2+1} \\ \hdashline W_4^0 & W_4^0 & W_4^{0+2} & W_4^{0+2} \\ W_4^0 & W_4^2 & W_4^{0+3} & W_4^{2+3} \end{array} \right] \begin{bmatrix} f_0 \\ f_4 \\ f_2 \\ f_6 \end{bmatrix}$$

$$\begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix} = \left[ \begin{array}{c} \left(f_0 W_4^0 + f_4 W_4^0\right) + \left(f_2 W_4^0 + f_6 W_4^0\right) W_4^0 \\ \left(f_0 W_4^0 + f_4 W_4^2\right) + \left(f_2 W_4^0 + f_6 W_4^2\right) W_4^1 \\ \hdashline \left(f_0 W_4^0 + f_4 W_4^0\right) + \left(f_2 W_4^0 + f_6 W_4^0\right) W_4^2 \\ \left(f_0 W_4^0 + f_4 W_4^2\right) + \left(f_2 W_4^0 + f_6 W_4^2\right) W_4^3 \end{array} \right] = \left[ \begin{array}{c} \left(f_0 W_2^0 + f_4 W_2^0\right) + \left(f_2 W_2^0 + f_6 W_2^0\right) W_4^0 \\ \left(f_0 W_2^0 + f_4 W_2^1\right) + \left(f_2 W_2^0 + f_6 W_2^1\right) W_4^1 \\ \hdashline \left(f_0 W_2^0 + f_4 W_2^0\right) + \left(f_2 W_2^0 + f_6 W_2^0\right) W_4^2 \\ \left(f_0 W_2^0 + f_4 W_2^1\right) + \left(f_2 W_2^0 + f_6 W_2^1\right) W_4^3 \end{array} \right]$$

$$= \left[ \begin{array}{c} J_0 + K_0 W_4^0 \\ J_1 + K_1 W_4^1 \\ \hdashline J_0 + K_0 W_4^2 \\ J_1 + K_1 W_4^3 \end{array} \right]$$

# 奇数行と偶数行に分ける

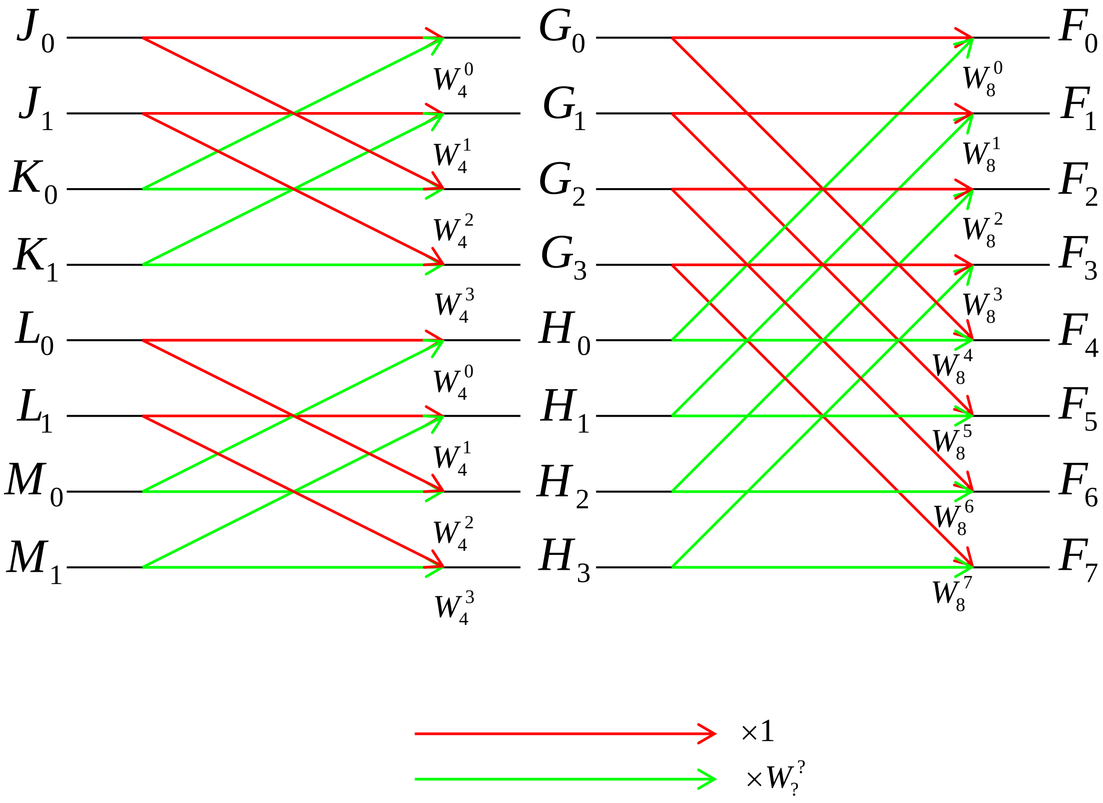
$$\begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ H_3 \end{bmatrix} = \begin{bmatrix} W_8^0 & W_8^0 & W_8^0 & W_8^0 \\ W_8^0 & W_8^2 & W_8^4 & W_8^6 \\ W_8^0 & W_8^4 & W_8^0 & W_8^4 \\ W_8^0 & W_8^6 & W_8^4 & W_8^2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_3 \\ f_5 \\ f_7 \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^0 & W_4^2 \\ W_4^0 & W_4^3 & W_4^2 & W_4^1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_3 \\ f_5 \\ f_7 \end{bmatrix}$$

$$\begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ H_3 \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^2 & W_4^1 & W_4^3 \\ W_4^0 & W_4^0 & W_4^2 & W_4^2 \\ W_4^0 & W_4^2 & W_4^3 & W_4^1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_5 \\ f_3 \\ f_7 \end{bmatrix} = \left[\begin{array}{cccc} W_4^0 & W_4^0 & W_4^{0+0} & W_4^{0+0} \\ W_4^0 & W_4^2 & W_4^{0+1} & W_4^{2+1} \\ \hdashline W_4^0 & W_4^0 & W_4^{0+2} & W_4^{0+2} \\ W_4^0 & W_4^2 & W_4^{0+3} & W_4^{2+3} \end{array}\right] \begin{bmatrix} f_1 \\ f_5 \\ f_3 \\ f_7 \end{bmatrix}$$

$$\begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ H_3 \end{bmatrix} = \left[\begin{array}{c} \left(f_1W_4^0 + f_5W_4^0\right) + \left(f_3W_4^0 + f_7W_4^0\right)W_4^0 \\ \left(f_1W_4^0 + f_5W_4^2\right) + \left(f_3W_4^0 + f_7W_4^2\right)W_4^1 \\ \hdashline \left(f_1W_4^0 + f_5W_4^0\right) + \left(f_3W_4^0 + f_7W_4^0\right)W_4^2 \\ \left(f_1W_4^0 + f_5W_4^2\right) + \left(f_3W_4^0 + f_7W_4^2\right)W_4^3 \end{array}\right] = \left[\begin{array}{c} \left(f_1W_2^0 + f_5W_2^0\right) + \left(f_3W_2^0 + f_7W_2^0\right)W_4^0 \\ \left(f_1W_2^0 + f_5W_2^1\right) + \left(f_3W_2^0 + f_7W_2^1\right)W_4^1 \\ \hdashline \left(f_1W_2^0 + f_5W_2^0\right) + \left(f_3W_2^0 + f_7W_2^0\right)W_4^2 \\ \left(f_1W_2^0 + f_5W_2^1\right) + \left(f_3W_2^0 + f_7W_2^1\right)W_4^3 \end{array}\right]$$

$$= \begin{bmatrix} L_0 + M_0W_4^0 \\ L_1 + M_1W_4^1 \\ L_0 + M_0W_4^2 \\ L_1 + M_1W_4^3 \end{bmatrix}$$

# バタフライ演算


$J_0$
$J_1$
$K_0$
$K_1$
$L_0$
$L_1$
$M_0$
$M_1$
$W_4^0$
$W_4^1$
$W_4^2$
$W_4^3$
$W_4^0$
$W_4^1$
$W_4^2$
$W_4^3$
$G_0$
$G_1$
$G_2$
$G_3$
$H_0$
$H_1$
$H_2$
$H_3$
$W_8^0$
$W_8^1$
$W_8^2$
$W_8^3$
$W_8^4$
$W_8^5$
$W_8^6$
$W_8^7$
$F_0$
$F_1$
$F_2$
$F_3$
$F_4$
$F_5$
$F_6$
$F_7$
×1
$\times W_?^?$

Ｎ＝２のDFT

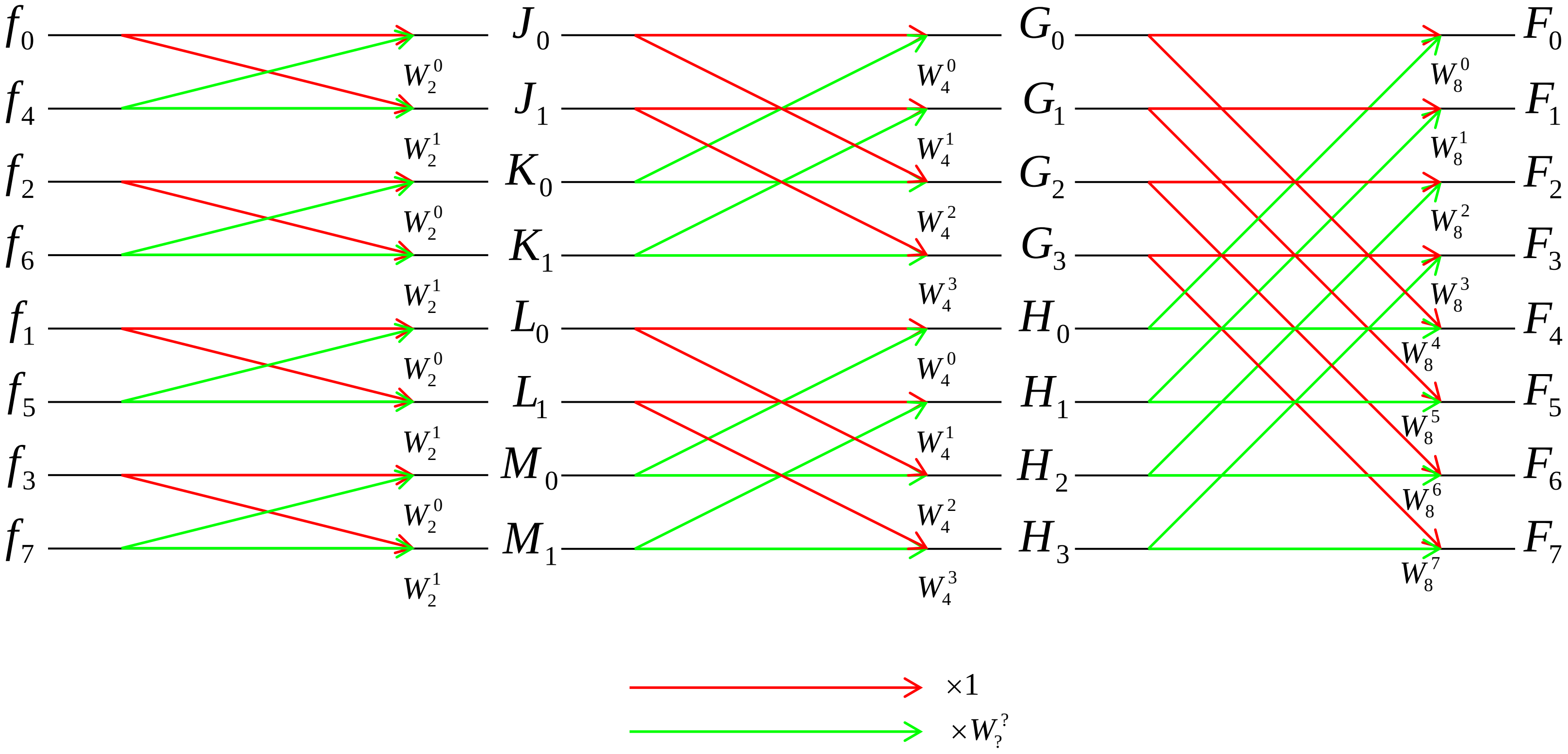
$$\begin{bmatrix} J_0 \\ J_1 \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 \\ W_4^0 & W_4^2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_4 \end{bmatrix} = \begin{bmatrix} W_2^0 & W_2^0 \\ W_2^0 & W_2^1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_4 \end{bmatrix}$$

$$\begin{bmatrix} K_0 \\ K_1 \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 \\ W_4^0 & W_4^2 \end{bmatrix} \begin{bmatrix} f_2 \\ f_6 \end{bmatrix} = \begin{bmatrix} W_2^0 & W_2^0 \\ W_2^0 & W_2^1 \end{bmatrix} \begin{bmatrix} f_2 \\ f_6 \end{bmatrix}$$

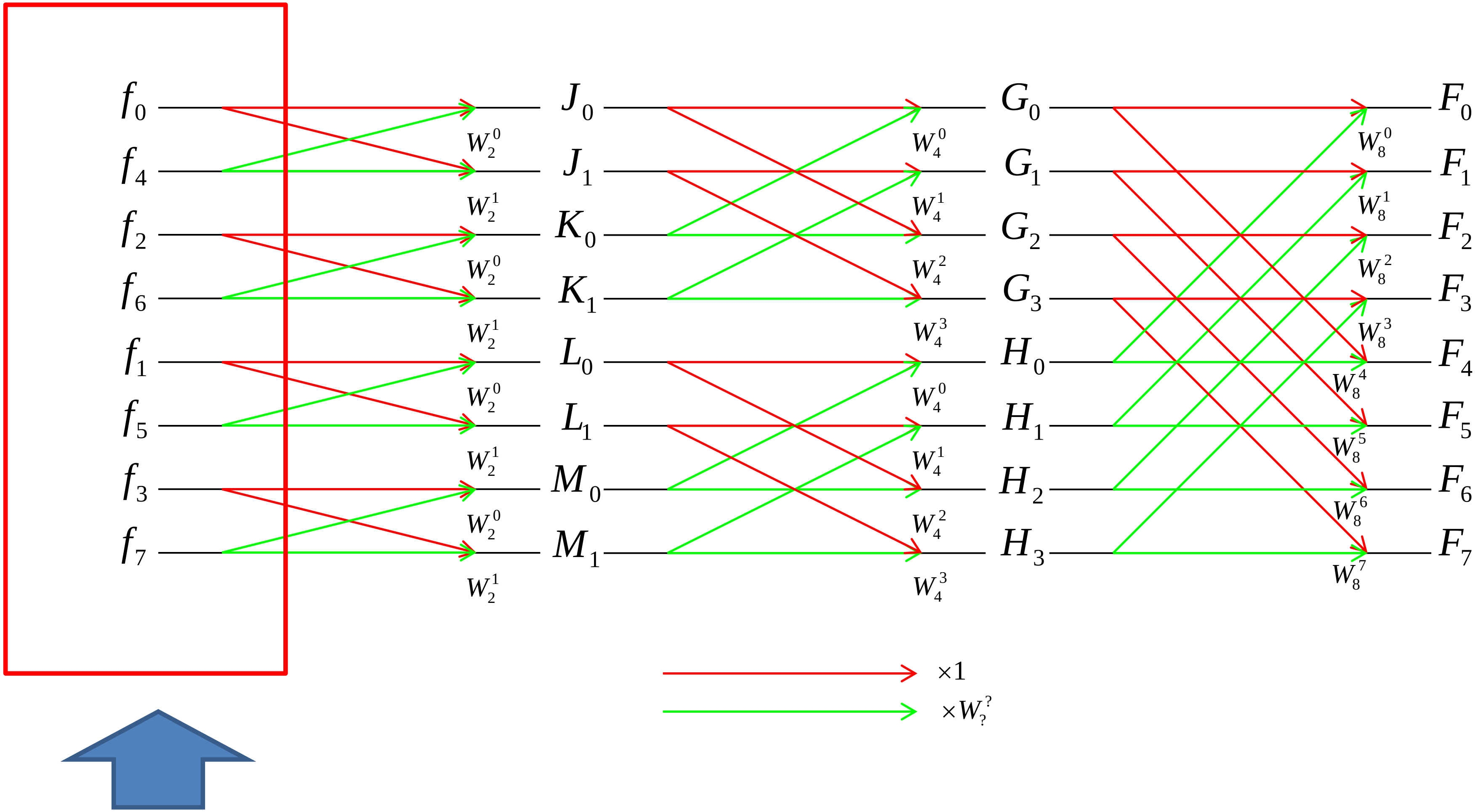
$$\begin{bmatrix} L_0 \\ L_1 \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 \\ W_4^0 & W_4^2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_5 \end{bmatrix} = \begin{bmatrix} W_2^0 & W_2^0 \\ W_2^0 & W_2^1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_5 \end{bmatrix}$$

$$\begin{bmatrix} M_0 \\ M_1 \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 \\ W_4^0 & W_4^2 \end{bmatrix} \begin{bmatrix} f_3 \\ f_7 \end{bmatrix} = \begin{bmatrix} W_2^0 & W_2^0 \\ W_2^0 & W_2^1 \end{bmatrix} \begin{bmatrix} f_3 \\ f_7 \end{bmatrix}$$

# バタフライ演算


$f_0$
$f_4$
$f_2$
$f_6$
$f_1$
$f_5$
$f_3$
$f_7$
$W_2^0$
$W_2^1$
$W_2^0$
$W_2^1$
$W_2^0$
$W_2^1$
$W_2^0$
$W_2^1$
$J_0$
$J_1$
$K_0$
$K_1$
$L_0$
$L_1$
$M_0$
$M_1$
$W_4^0$
$W_4^1$
$W_4^2$
$W_4^3$
$W_4^0$
$W_4^1$
$W_4^2$
$W_4^3$
$G_0$
$G_1$
$G_2$
$G_3$
$H_0$
$H_1$
$H_2$
$H_3$
$W_8^0$
$W_8^1$
$W_8^2$
$W_8^3$
$W_8^4$
$W_8^5$
$W_8^6$
$W_8^7$
$F_0$
$F_1$
$F_2$
$F_3$
$F_4$
$F_5$
$F_6$
$F_7$
$\times 1$
$\times W_?^?$

# 全体の信号の流れ



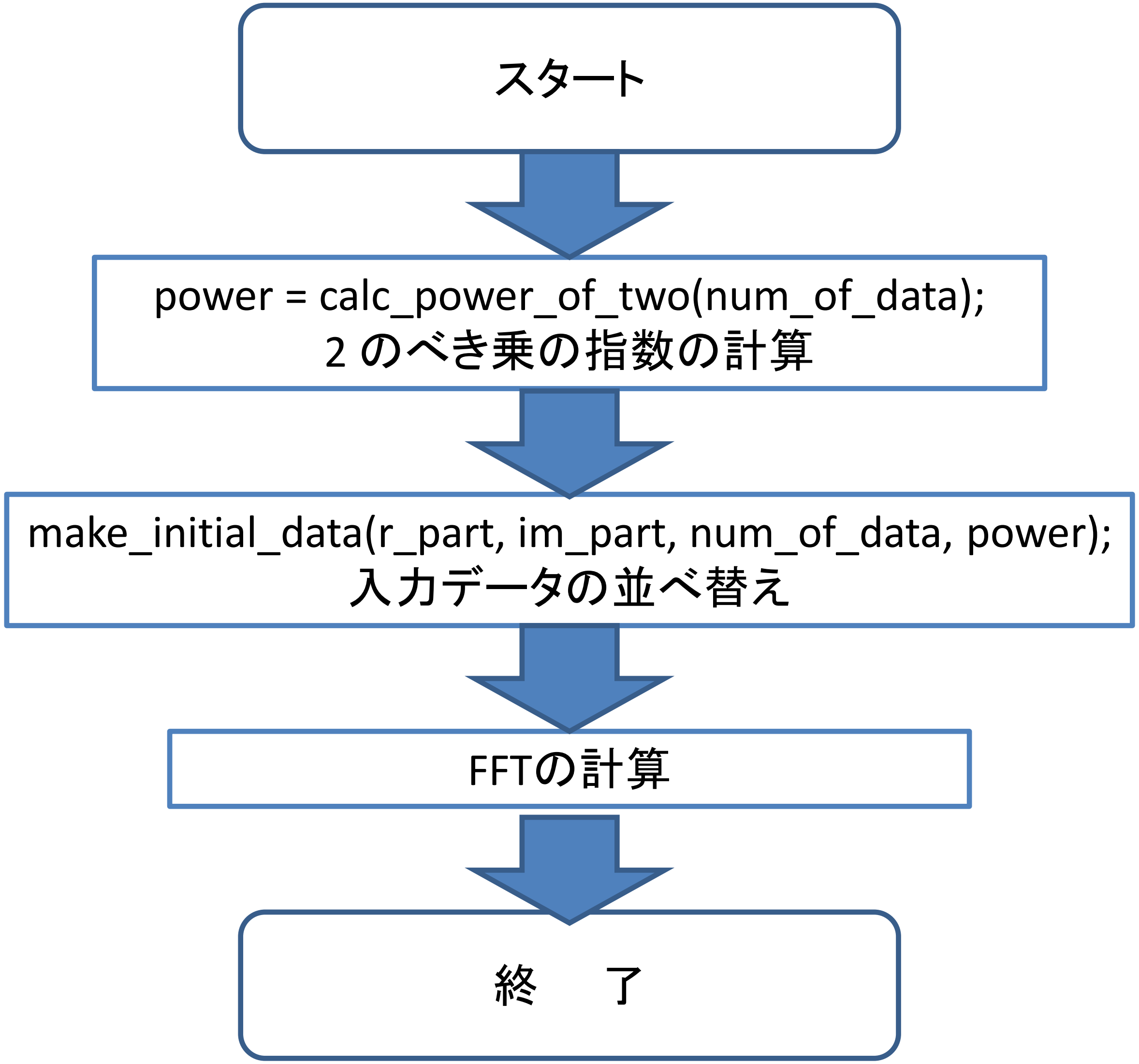
入力データをこの順番に並べ替えると演算がしやすくなる

# 関数 FFT1の説明

- １次元（例えば時系列データ）のFFTを計算する関数

- 引数
  - double  *r_part  ：  入力データの実部
  - double *im_part  :入力データの虚部
  - int num_of_data  :入力データの数
  - int flag  ：  1であればFFT、-1であれば逆FFT

# 全体の流れ（順変換のみに着目）



スタート

power = calc_power_of_two(num_of_data);

2のべき乗の指数の計算

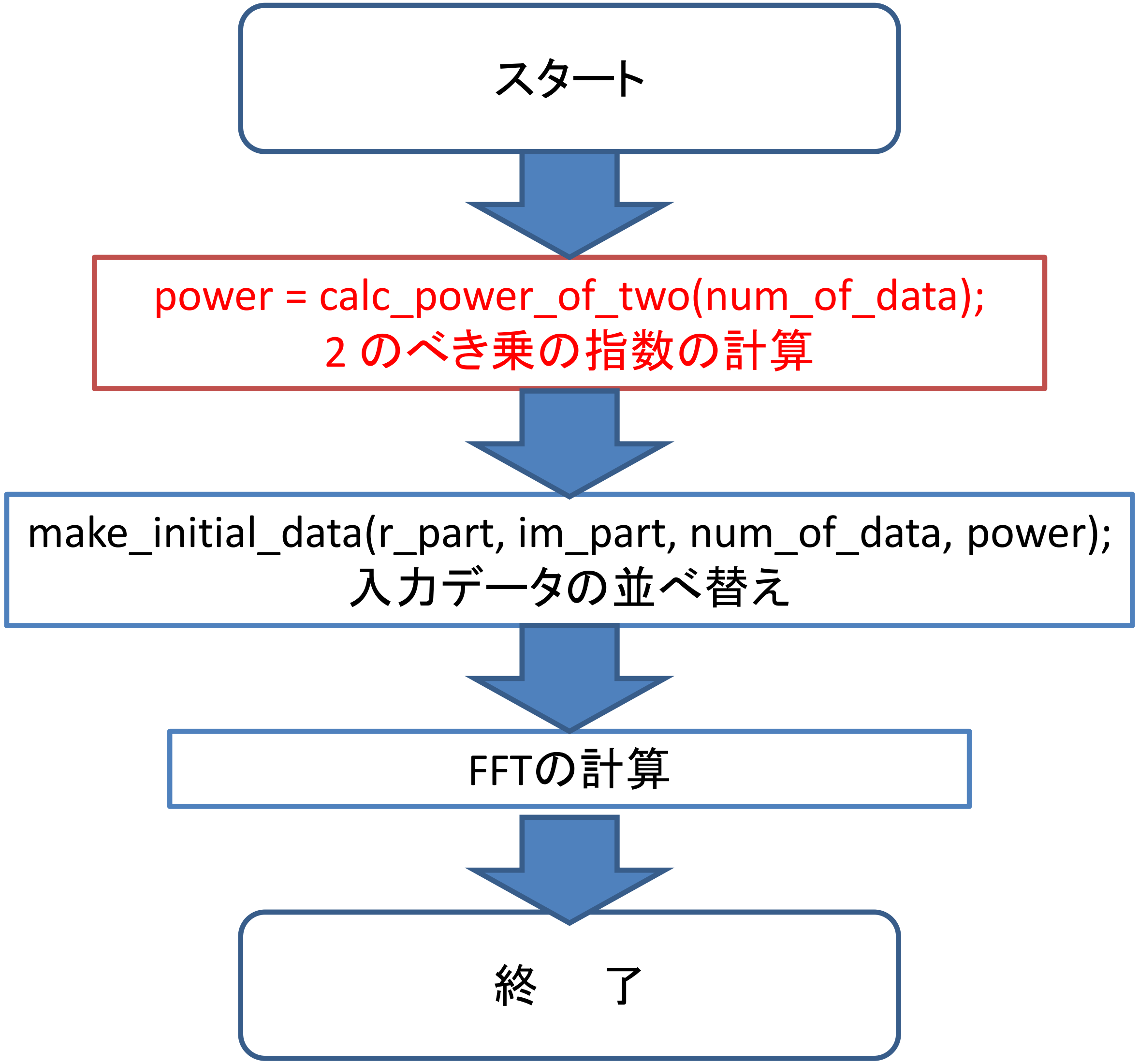
make_initial_data(r_part, im_part, num_of_data, power);

入力データの並べ替え

FFTの計算

終　了

# 全体の流れ（順変換のみに着目）



スタート

power = calc_power_of_two(num_of_data);
2 のべき乗の指数の計算

make_initial_data(r_part, im_part, num_of_data, power);
入力データの並べ替え

FFTの計算

終　了

# 2のべき乗の指数の計算

```
int calc_power_of_two(int number)
{
        int power = 0;
        while (number != 1) {
                power++; number = number /2
        }
        return (power);
}
```

# 例:number=8

```
int calc_power_of_two(8)
{
        int power = 0;
        while (8 != 1) {
                power++; number = number /2;
        }
        return (power);
}
```

# 例：number=8

```
int calc_power_of_two(8)
{
        int power = 0;
        while (8 != 1) {
                power=1; number = 4;
        }
        return (power);
}
```

# 例：number=8

```
int calc_power_of_two(8)
{
        int power = 0;
        while (4 != 1) {
                power=2; number = 2;
        }
        return (power);
}
```

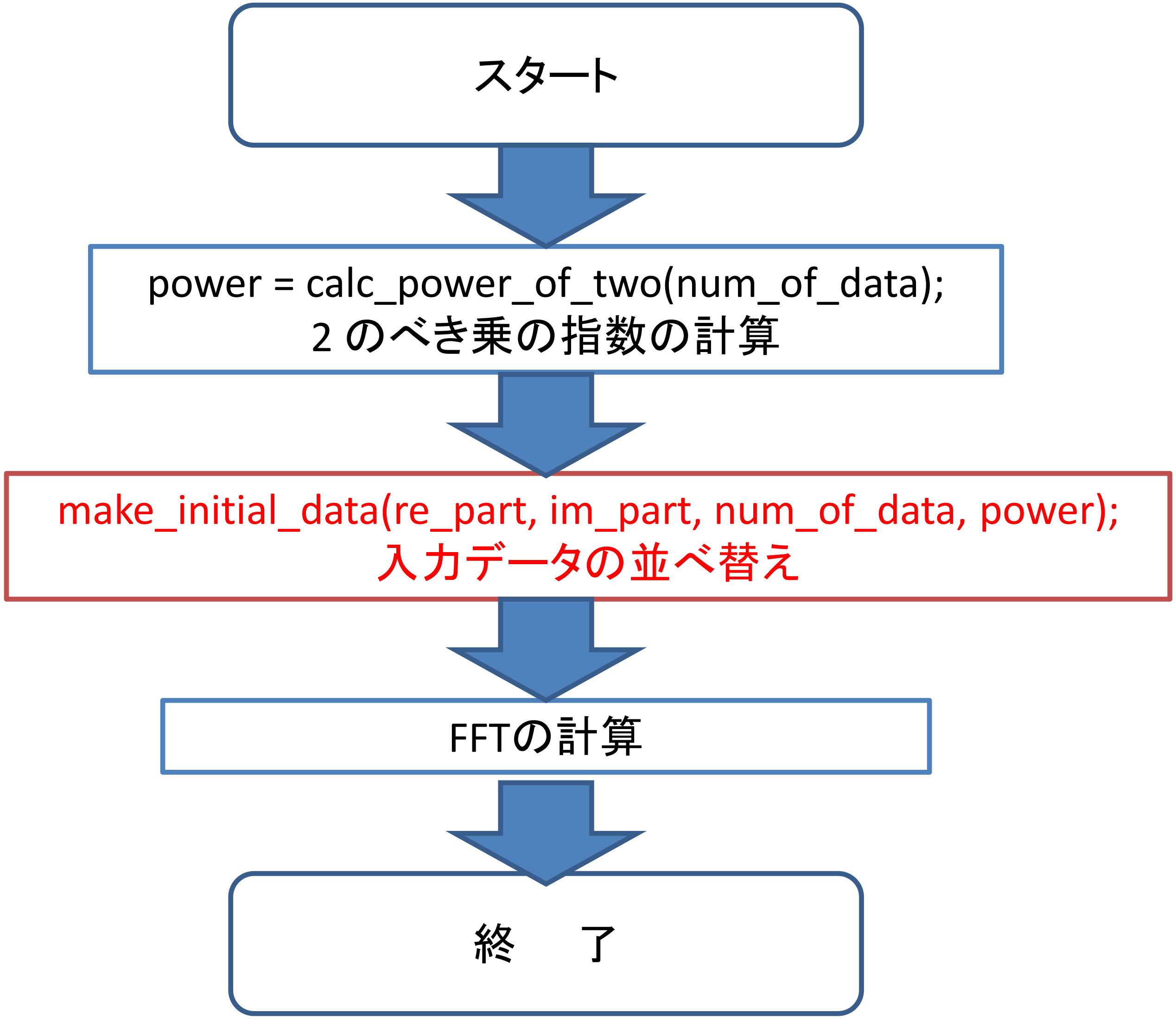
# 例：number=8

```
int calc_power_of_two(8)
{
        int power = 0;
        while (2 != 1) {
                power=3; number = 1;
        }
        return (power);
}
```

# 例 : number=8=$2^3$

```
int calc_power_of_two(8)
{
        int power = 0;
        while (1 != 1) {
                power=3; number = 1;
        }
        return (3);
}
```

# 全体の流れ（順変換のみに着目）



スタート

power = calc_power_of_two(num_of_data);
2 のべき乗の指数の計算

make_initial_data(re_part, im_part, num_of_data, power);
入力データの並べ替え

FFTの計算

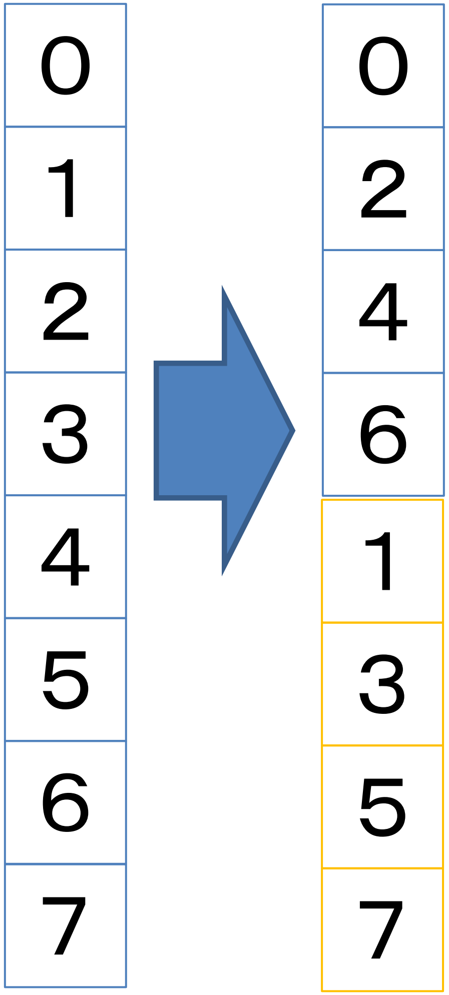
終　了

# 並べ替えアルゴリズム

| |
|---|
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |

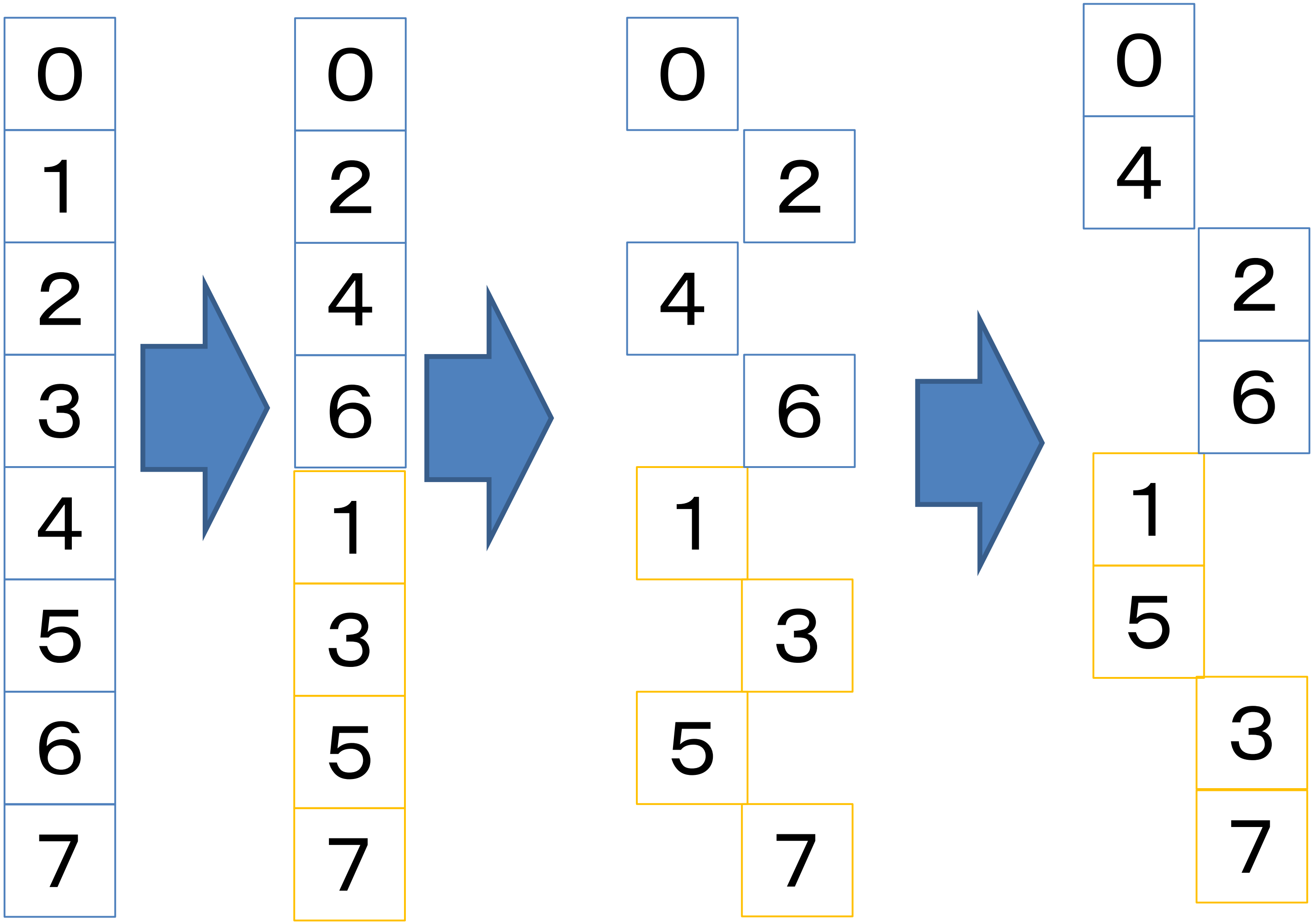
奇数行と偶数行に分ける

# 並べ替えアルゴリズム



| | |
|---|---|
| 0 | 0 |
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | 1 |
| 5 | 3 |
| 6 | 5 |
| 7 | 7 |

# 並べ替えアルゴリズム



# make_initial_data(double *r_part, double *im_part, int num_of_data, int power)

```
dft= num_of_data;
for (i = 1; i < power; i++) {
        new_ptr = 0; offset = 0;
        while (new_ptr < num_of_data) {
                ptr = 0;
                while (ptr < dft) {
                                re_buf[new_ptr] = *(re_part + offset + ptr);
                                im_buf[new_pntr] = *(im_part +offset + ptr);
                                new_ptr++;
                                ptr = ptr + 2;
                                if (ptr == dft) ptr = 1;
                }
                offset = offset + dft;
        } /* result of calculation is copied into arrays */
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/ 2;
 }
```

三重ループ

このループについて

# make_initial_data関数の引数

- make_initial_data(double *re_part, double *im_part, int num_of_data, int power)

- *re_part, *im_part : 入力データの実部・虚部

- num_of_data : 入力データ数

- power : 2 のべき乗の指数( num_of_data = 8であれば3)

# 例：データ数 num_of_data=8=$2^3$

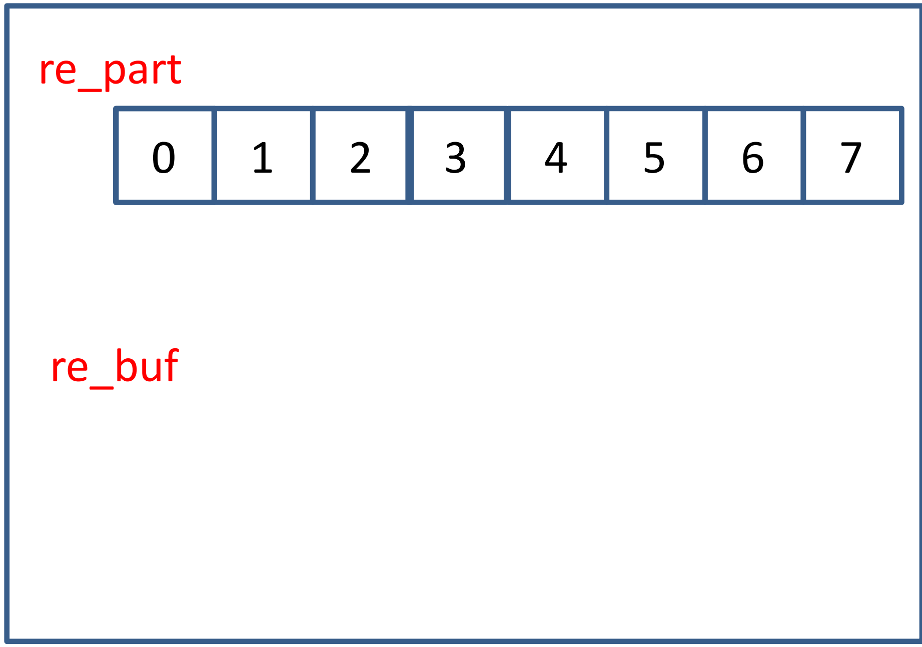
```
dft= num_of_data;
for (i = 1; i < power; i++) {
        new_pntr = 0; offset = 0;
        while (new_ptr < num_of_data) {
                pntr = 0;
                while (ptr < DFT) {
                                re_buf [new_ptr] = *(re_part + offset + ptr);
                                re_buf [new_ptr] = *(im_part +offset + ptr);
                                new_ptr++;
                                ptr = ptr + 2;
                                if (ptr == dft) ptr = 1;
                }
                offset = offset + dft;
        } /* result of calculation is copied into arrays */
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/ 2;
}
```

calc_power_of_two関数で既に求めている

# 例：データ数8

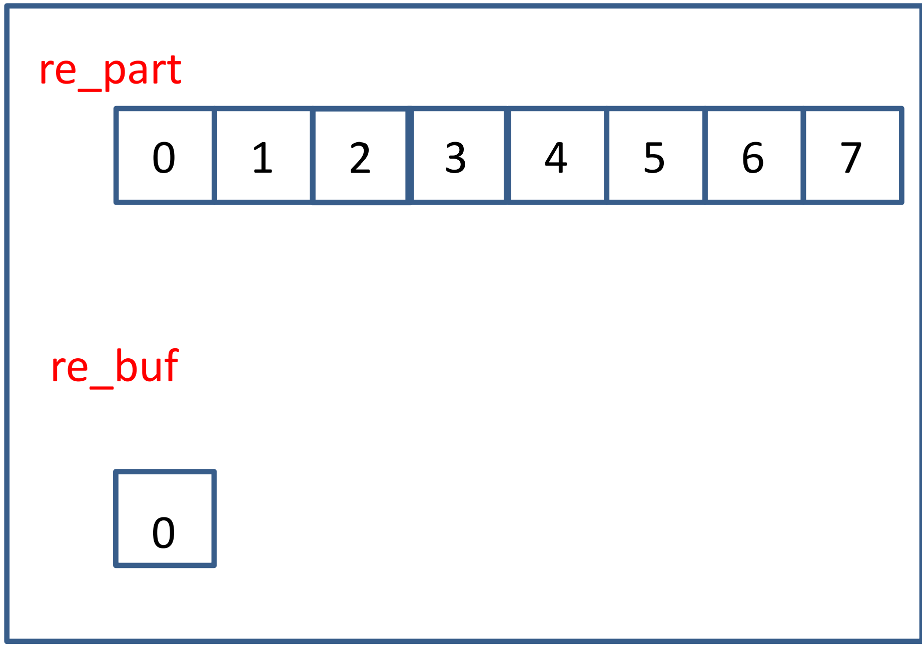
```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (0 < 8) {
                                re_buf[0] = *(re_part + 0+ 0);
                                im_buf[0] = *(im_part +0+0);
                                new_ptr= 1
                                ptr=2
                                if (2== 8) pntr = 1;
                }
                offset = offset + dft;
        } /* result of calculation is copied into arrays */
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/ 2;
}
```

re_part[0]と同意



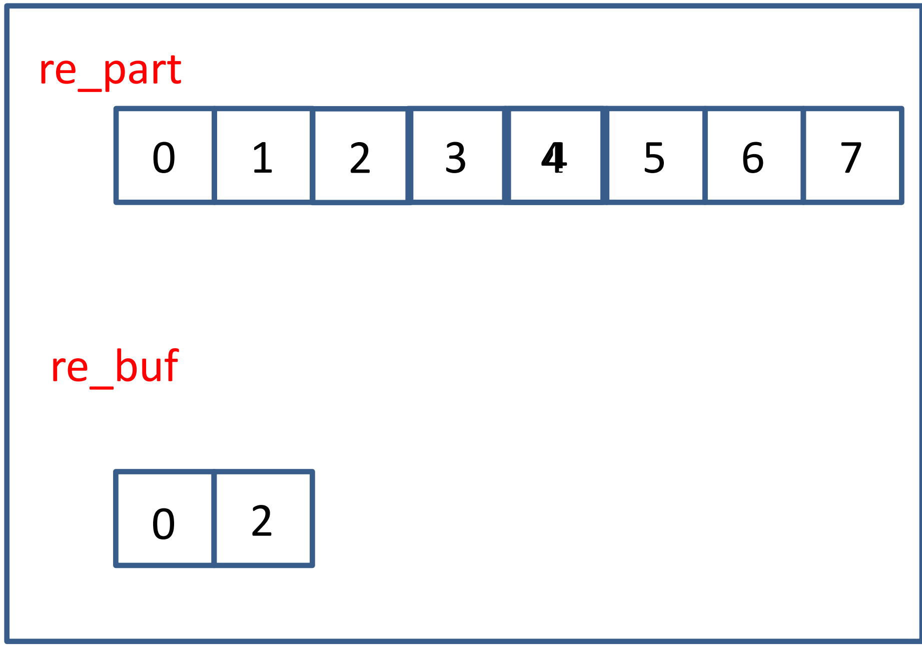
# 例 : データ数8

```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (2 < 8) {
                                re_buf[1] = *(re_part + 0+ 2);
                                im_buf[1] = *(im_part +0+2);
                                new_ptr= 2
                                ptr=4
                                if (4== 8) ptr = 1;
                }
                offset = offset + dft;
        } /* result of calculation is copied into arrays */
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/ 2;
 }
```



# 例：データ数8

```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (4< 8) {
                                re_buf[2] = *(re_part + 0+ 4);
                                im_buf[2] = *(im_part +0+4);
                                new_ptr= 3
                                ptr=6
                                if (6== 8) ptr = 1;
                }
                offset = offset + dft;
        } /* result of calculation is copied into arrays */
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft = dft / 2;
 }
```



re_part

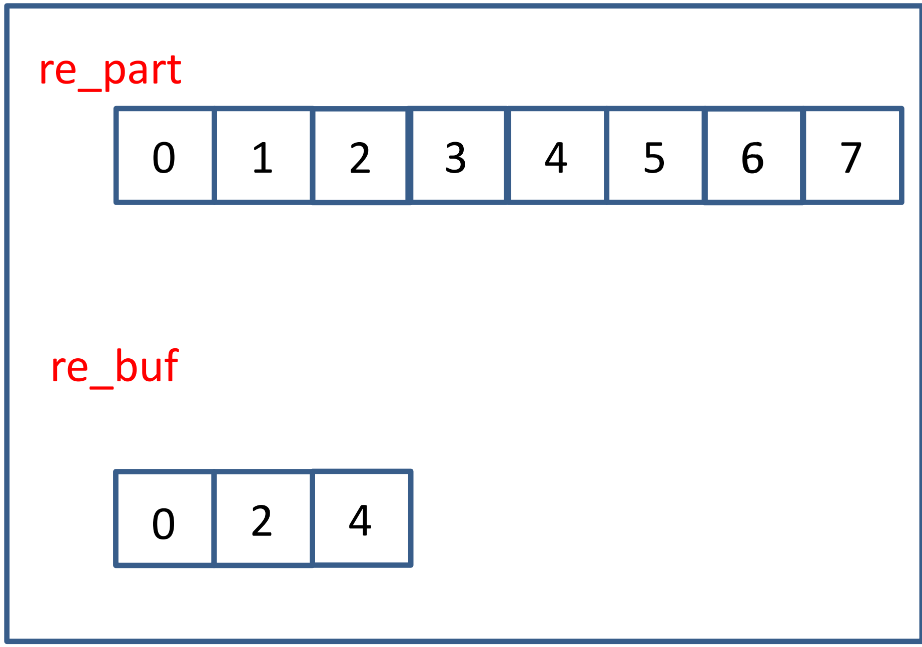
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|

re_buf

| 0 | 2 |
|---|---|

# 例：データ数8

```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (6< 8) {
                                re_buf[3] = *(re_part + 0+ 6);
                                im_buf[3] = *(im_part +0+6);
                                new_ptr= 4
                                ptr=8
                                if (8== 8) ptr = 1;
                }
                offset = offset + dft;
        } /* result of calculation is copied into arrays */
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/ 2;
}
```



re_part

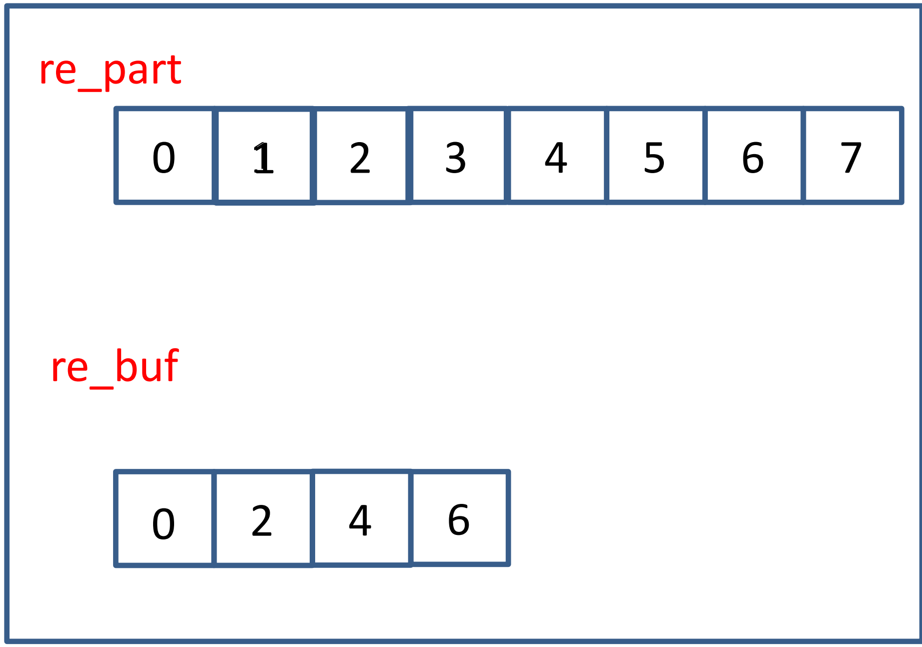
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|

re_buf

| 0 | 2 | 4 |
|---|---|---|

# 例：データ数8

```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (1< 8) {
                                re_buf[4] = *(re_part + 0+ 1);
                                im_buf[4] = *(im_part +0+1);
                                new_ptr= 5
                                ptr=3
                                if (3== 8) ptr = 1;
                }
                offset = offset + dft;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft / 2;
}
```



re_part

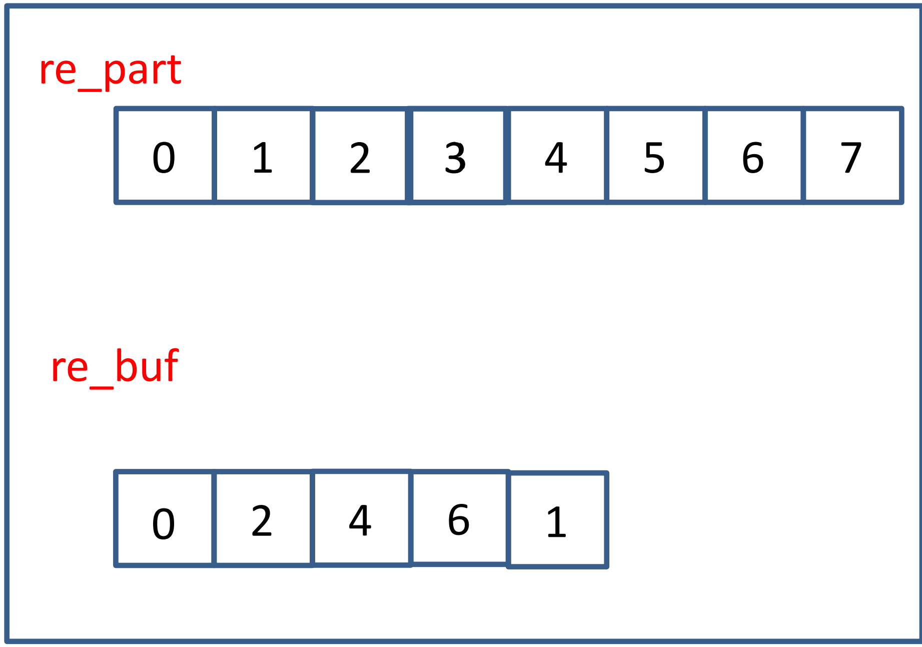
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

re_buf

| | | | |
|---|---|---|---|
| 0 | 2 | 4 | 6 |

# 例：データ数8

```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (3< 8) {
                                re_buf[5] = *(re_part + 0+ 3);
                                im_buf[5] = *(im_part +0+3);
                                new_ptr= 6
                                ptr=5
                                if (5== 8) ptr = 1;
                }
                offset = offset + dft;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft / 2;
}
```



re_part

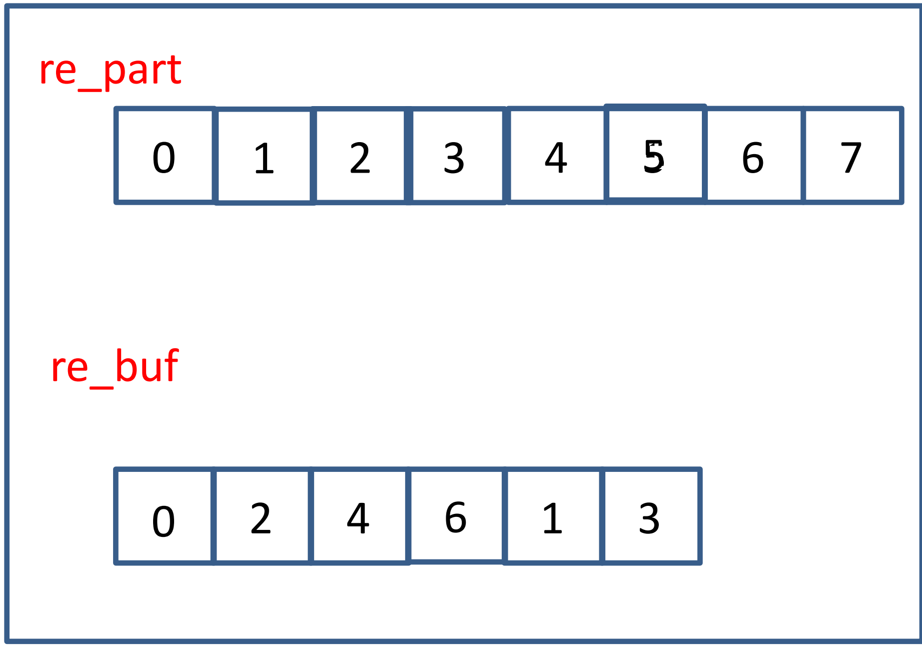
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|

re_buf

| 0 | 2 | 4 | 6 | 1 |
|---|---|---|---|---|

# 例：データ数8

```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                pntr = 0;
                while (5< 8) {
                                re_buf[5] = *(re_part + 0+ 5);
                                im_buf[5] = *(im_part +0+5);
                                new_ptr= 7
                                ptr=7
                                if (7== 8) ptr = 1;
                }
                offset = offset + dft;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft / 2;
}
```



re_part

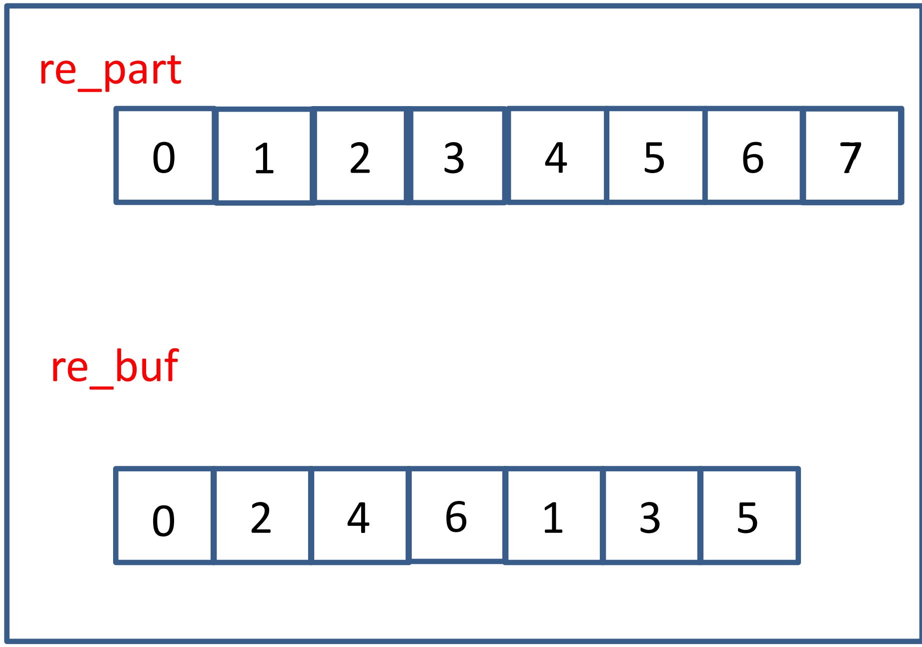
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|

re_buf

| 0 | 2 | 4 | 6 | 1 | 3 |
|---|---|---|---|---|---|

# 例 : データ数8

```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                pntr = 0;
                while (7< 8) {
                                re_buf[7] = *(re_part + 0+ 7);
                                im_buf[7] = *(im_part +0+7);
                                new_ptr= 8
                                ptr=9
                                if (9== 8) ptr = 1;
                }
                offset = offset + dft;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft / 2;
}
```



re_part

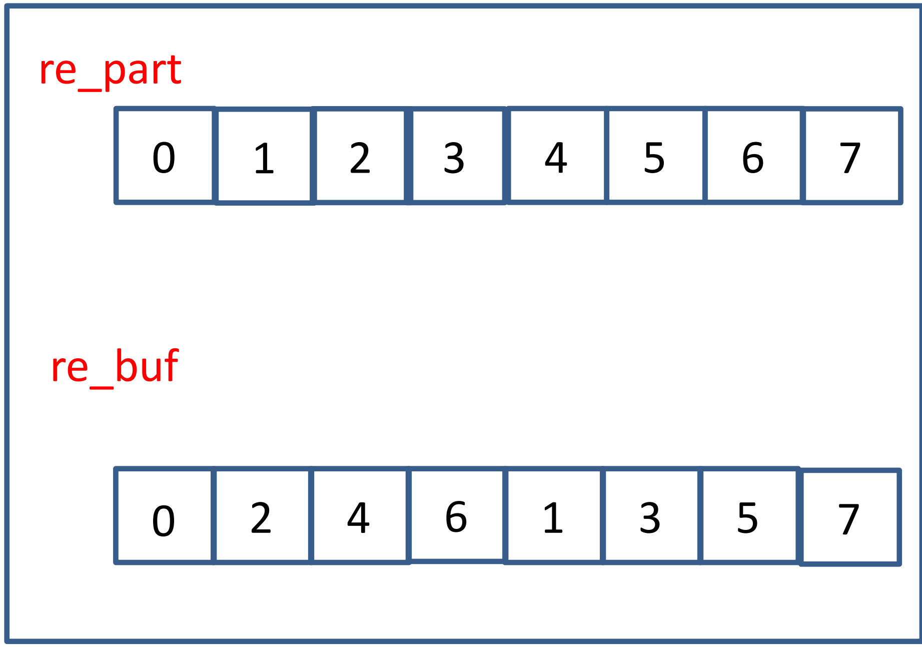
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|

re_buf

| 0 | 2 | 4 | 6 | 1 | 3 | 5 |
|---|---|---|---|---|---|---|

# 例：データ数8

```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (8< 8) {
                ptr = 0;
                while (7< 8) {
                                re_buf[7] = *(re_part + 0+ 7);
                                im_buf[7] = *(im_part +0+7);
                                new_ptr= 8
                                ptr=9
                                if (9== 8) ptr = 1;
                }
                offset = 8;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft / 2;
}
```



re_part

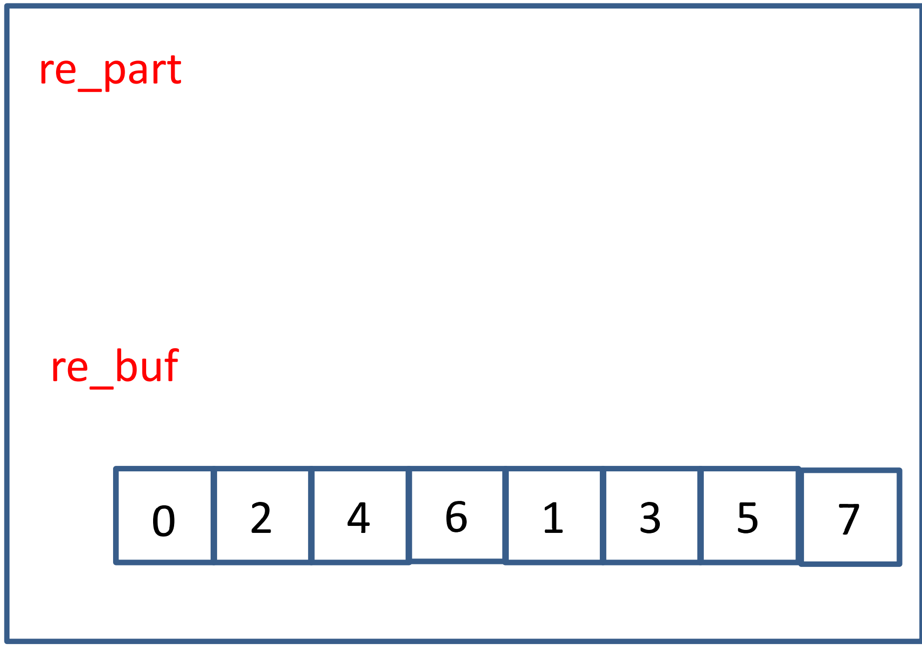
0 1 2 3 4 5 6 7

re_buf

0 2 4 6 1 3 5 7

# 例：データ数8

```
dft = num_of_data=8;
for (i = 1; i < 3; i++) {
        new_ptr = 0; offset = 0;
        while (8< 8) {
                ptr = 0;
                while (7< 8) {
                                re_buf[7] = *(re_part + 0+ 7);
                                im_buf[7] = *(im_part +0+7);
                                new_ptr= 8
                                ptr=9
                                if (9== 8) ptr = 1;
                }
                offset = 8;
        }
        for (j = 0; j < 8; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= 8/ 2;= 4
 }
```



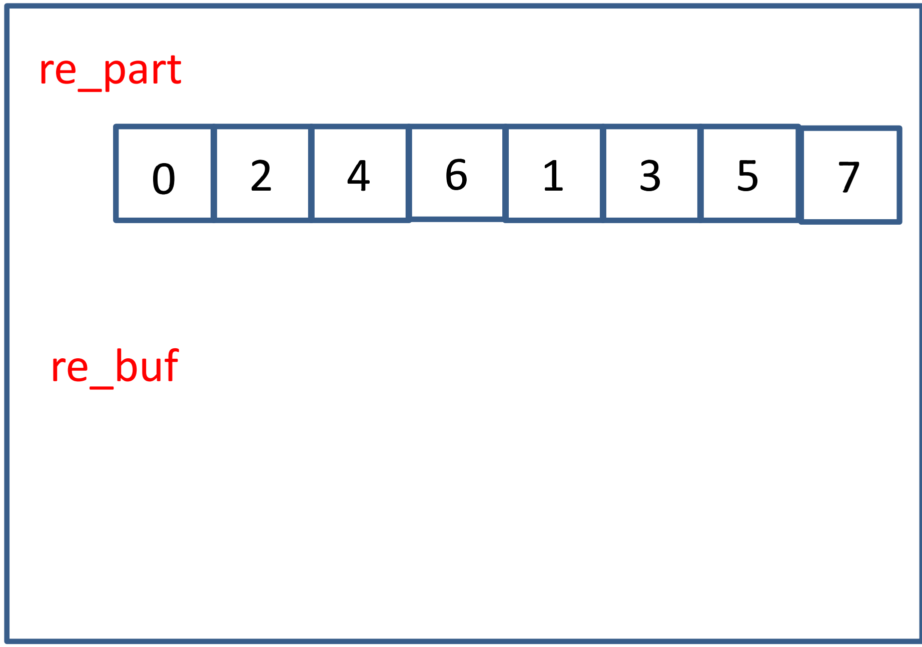
re_part

re_buf

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (0< 4) {
                                re_buf[new_ptr] = *(re_part + offset + ptr);
                                im_buf[new_ptr] = *(im_part +offset + ptr);
                                new_ptr++;
                                ptr = ptr + 2;
                                if (ptr == dft) ptr = 1;
                }
                offset = offset+4;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/2
}
```



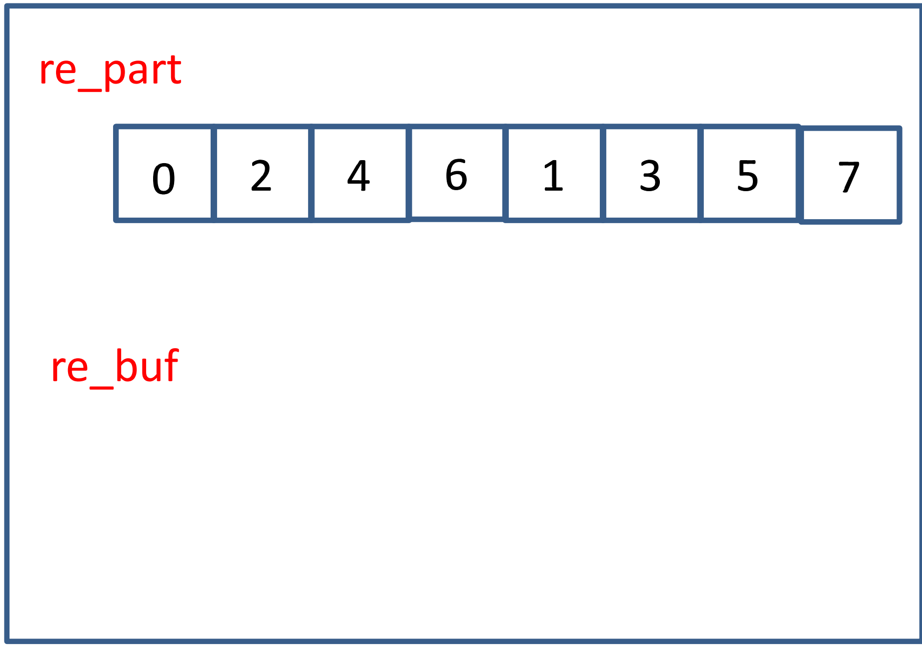
re_part

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |

re_buf

# 例 : データ数8

```
for (i = 2; i < 3; i++ i=2) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (0< 4) {
                                re_buf[[0] = *(re_part + 0 + 0);
                                im_buf[[0] = *(im_part +0+ 0);
                                new_ptr=1
                                ptr = 2
                                if (2== 4) ptr = 1;
                }
                offset = 4;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/2
}
```



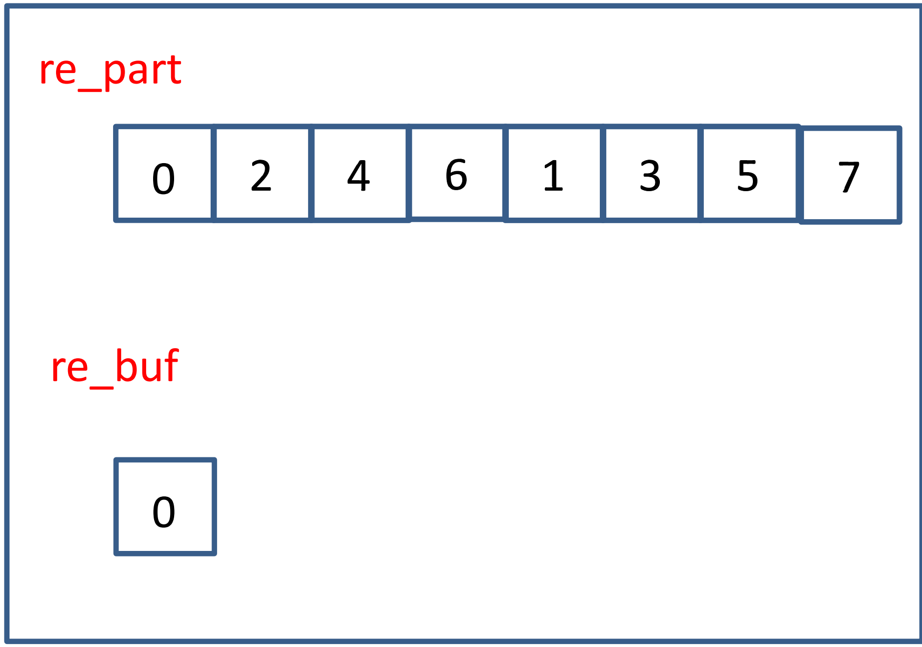
re_part

| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |
|---|---|---|---|---|---|---|---|

re_buf

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (2< 4) {
                                re_buf[[1] = *(re_part + 0 + 2);
                                im_buf[[1] = *(im_part +0+ 2);
                                new_ptr=2
                                ptr = 4
                                if (4== 4) ptr = 1;
                }
                offset = 4;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/2
 }
```



re_part

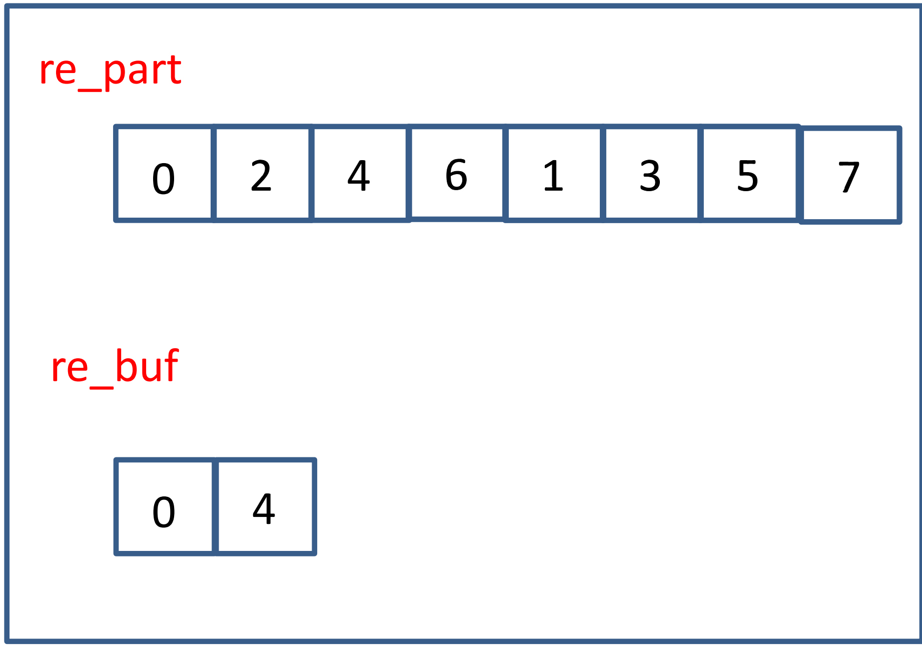
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |

re_buf

| |
|---|
| 0 |

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (1< 4) {
                                re_buf[[2] = *(re_part + 0 + 1);
                                im_buf[[2] = *(im_part +0+ 1);
                                new_ptr=3
                                ptr = 3
                                if (3== 4) ptr = 1;
                }
                offset =4;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/2
}
```



re_part

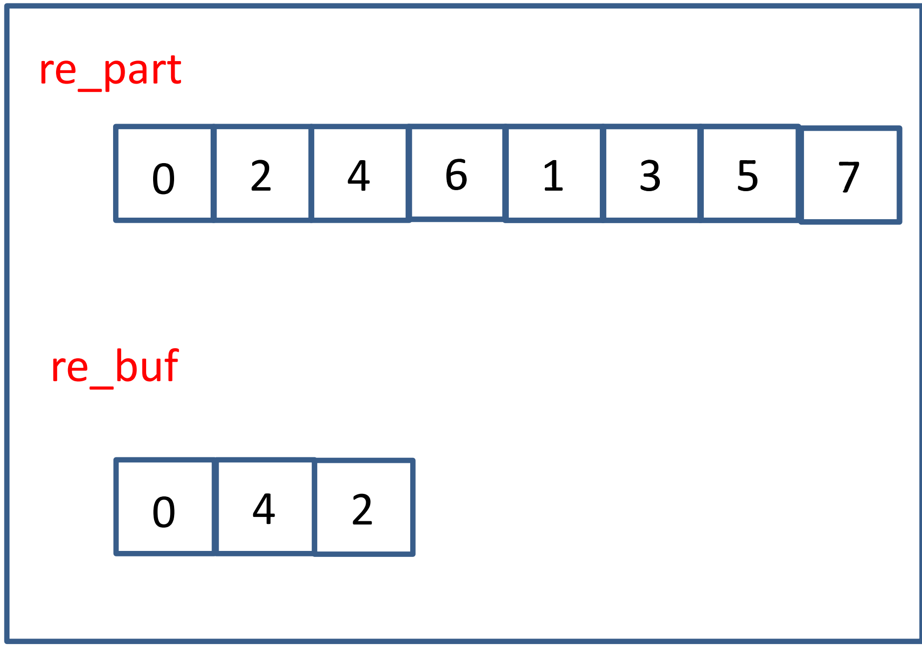
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |

re_buf

| | |
|---|---|
| 0 | 4 |

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
        new_ptr = 0; offset = 0;
        while (0< 8) {
                ptr = 0;
                while (3< 4) {
                                re_buf[[3] = *(re_part + 0 + 3);
                                im_buf[[3] = *(im_part +0+ 3);
                                new_ptr=4
                                ptr = 5
                                if (5== 4) ptr = 1;
                }
                offset = offset+4;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/2
 }
```



re_part

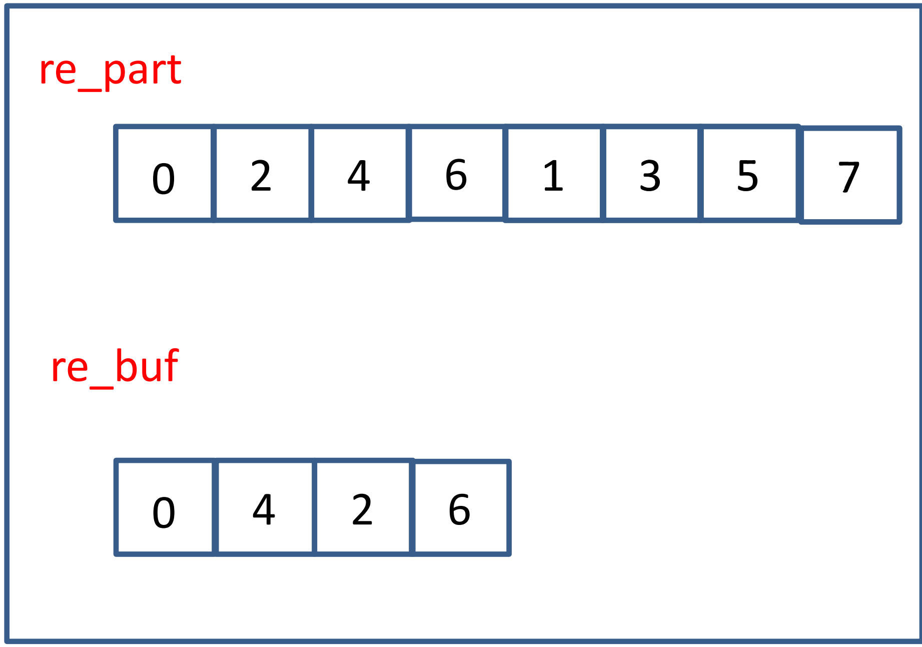
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |

re_buf

| | | |
|---|---|---|
| 0 | 4 | 2 |

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
        new_ptr = 0; offset = 0;
        while (4< 8) {
                ptr = 0;
                while (3< 4) {
                                re_buf[[3] = *(re_part + 0 + 3);
                                im_buf[[3] = *(im_part +0+ 3);
                                new_ptr=4
                                ptr = 5
                                if (5== 4) ptr = 1;
                }
                offset = offset+4;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/2
}
```



re_part

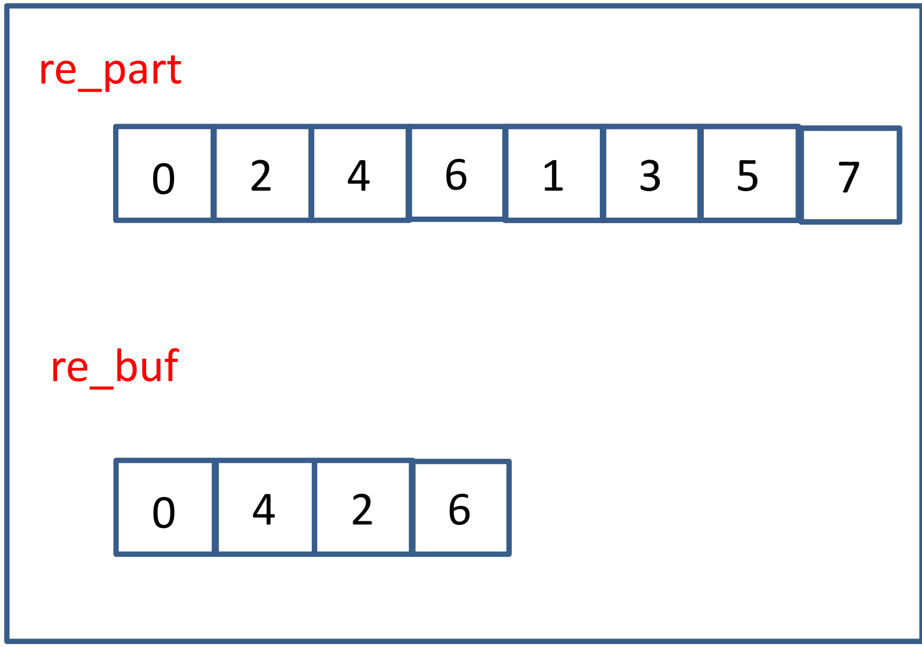
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |

re_buf

| | | | |
|---|---|---|---|
| 0 | 4 | 2 | 6 |

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
        new_ptr = 0; offset = 0;
        while (4< 8) {
                ptr = 0;
                while (0< 4) {
                                re_buf[[4] = *(re_part + 4+ 0);
                                im_buf[[4] = *(im_part +4+ 0);
                                new_ptr=5
                                ptr = 2
                                if (2== 4) ptr = 1;
                }
                offset = offset+4;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/2
}
```



re_part

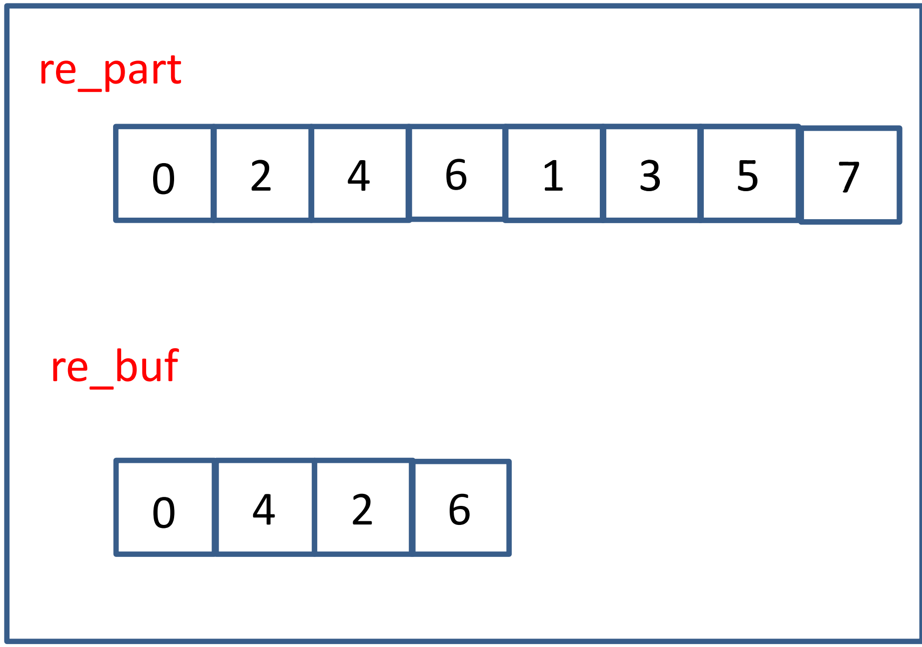
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |
|---|---|---|---|---|---|---|---|

re_buf

| 0 | 4 | 2 | 6 |
|---|---|---|---|

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
    new_ptr = 0; offset = 0;
    while (4< 8) {
        ptr = 0;
        while (0< 4) {
                re_buf[[4] = *(re_part + 4+ 0);
                im_buf[[4] = *(im_part +4+ 0);
                new_ptr=5
                ptr = 2
                if (2== 4) ptr = 1;
        }
        offset = offset+4;
    }
    for (j = 0; j < num_of_data; j++) {
        *(re_part + j) = re_buf[j];
        *(im_part + j) = im_buf[j];
    }
    dft= dft/2
}
```



re_part

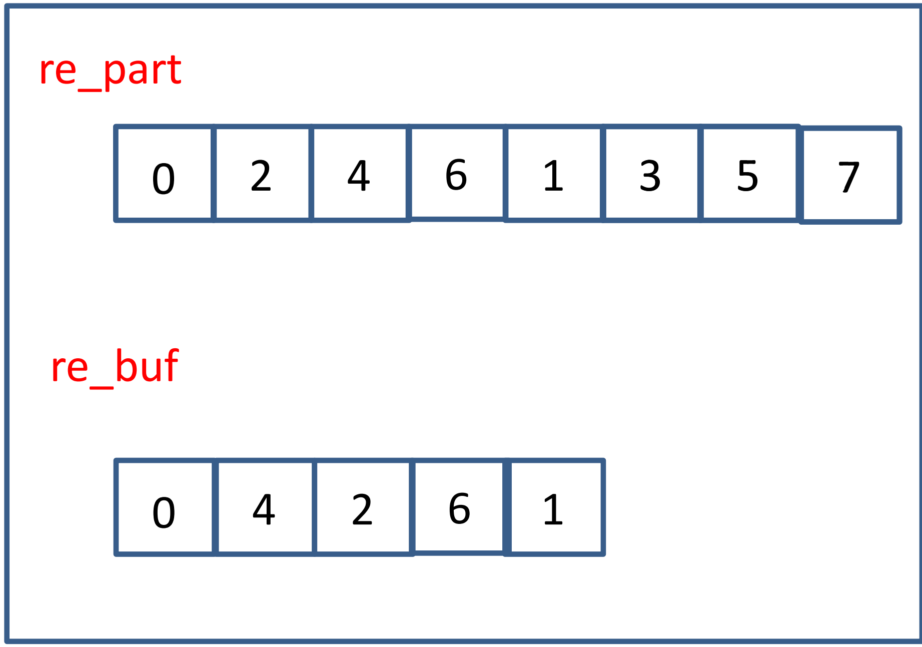
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |
|---|---|---|---|---|---|---|---|

re_buf

| 0 | 4 | 2 | 6 |
|---|---|---|---|

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
        new_ptr = 0; offset = 0;
        while (4< 8) {
                ptr = 0;
                while (2< 4) {
                                re_buf[[5] = *(re_part + 4+ 2);
                                im_buf[[5] = *(im_part +4+ 2);
                                new_ptr=6
                                ptr = 4
                                if (4== 4) ptr = 1;
                }
                offset = offset+4;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= dft/2
}
```



re_part

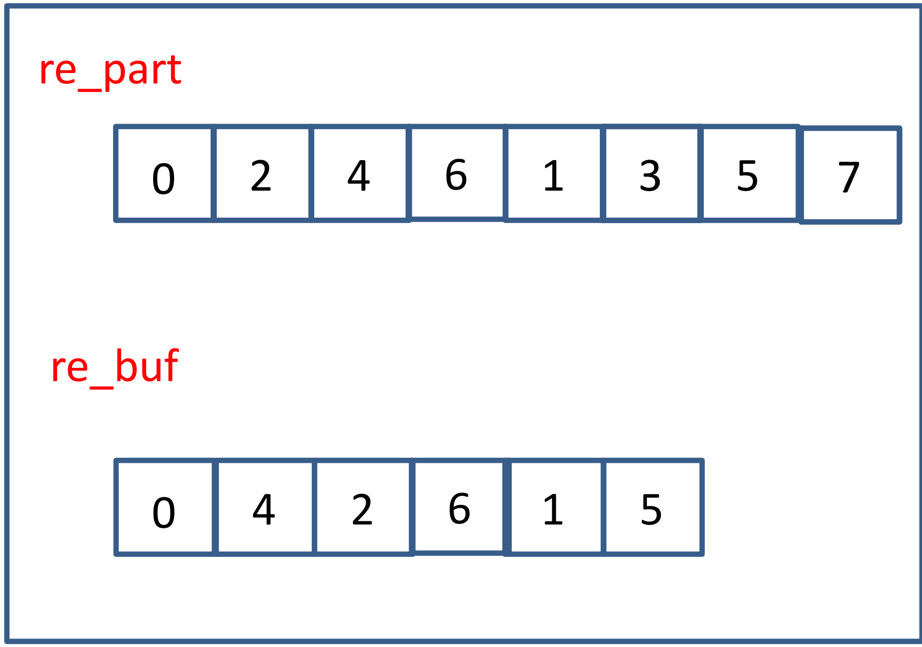
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |
|---|---|---|---|---|---|---|---|

re_buf

| 0 | 4 | 2 | 6 | 1 |
|---|---|---|---|---|

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
    new_ptr = 0; offset = 0;
    while (4< 8) {
        ptr = 0;
        while (1< 4) {
                re_buf[[6] = *(re_part + 4+ 1);
                im_buf[[6] = *(im_part +4+ 1);
                new_ptr=7
                ptr = 3
                if (3== 4) ptr = 1;
        }
        offset = offset+4;
    }
    for (j = 0; j < num_of_data; j++) {
        *(re_part + j) = re_buf[j];
        *(im_part + j) = im_buf[j];
    }
    dft= dft/2
}
```



re_part

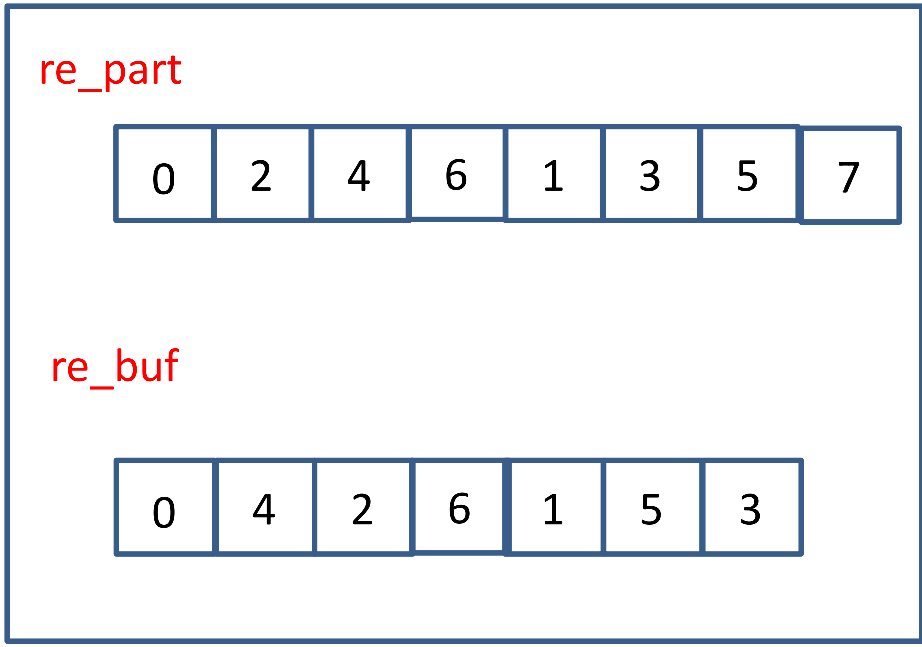
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |

re_buf

| | | | | | |
|---|---|---|---|---|---|
| 0 | 4 | 2 | 6 | 1 | 5 |

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
    new_ptr = 0; offset = 0;
    while (4< 8) {
        ptr = 0;
        while (3< 4) {
                re_buf[[7] = *(re_part + 4+ 3);
                im_buf[[7] = *(im_part +4+ 3);
                new_ptr=8
                ptr = 5
                if (5== 4) ptr = 1;
        }
        offset = 4;
    }
    for (j = 0; j < num_of_data; j++) {
        *(re_part + j) = re_buf[j];
        *(im_part + j) = im_buf[j];
    }
    dft= dft/2
}
```



re_part

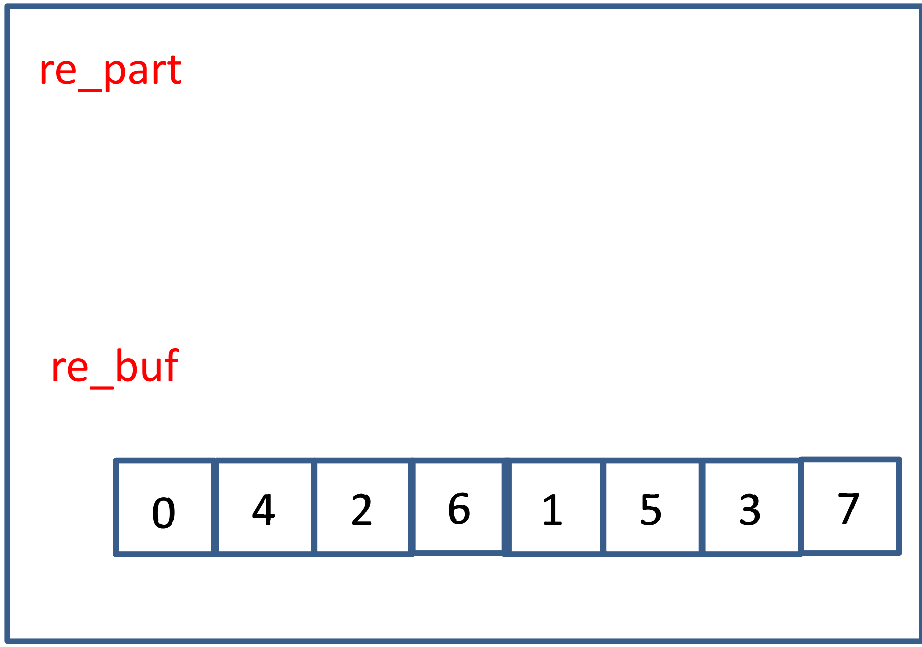
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |

re_buf

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 4 | 2 | 6 | 1 | 5 | 3 |

# 例：データ数8

```
for (i = 2; i < 3; i++ i=2) {
        new_ptr = 0; offset = 0;
        while (8< 8) {
                ptr = 0;
                while (5< 4) {
                                re_buf[[7] = *(re_part + 4+ 3);
                                im_buf[[7] = *(im_part +4+ 3);
                                new_ptr=8
                                ptr = 5
                                if (5== 4) ptr = 1;
                }
                offset = 4;
        }
        for (j = 0; j < num_of_data; j++) {
                *(re_part + j) = re_buf[j];
                *(im_part + j) = im_buf[j];
        }
        dft= 4/2=2
 }
```

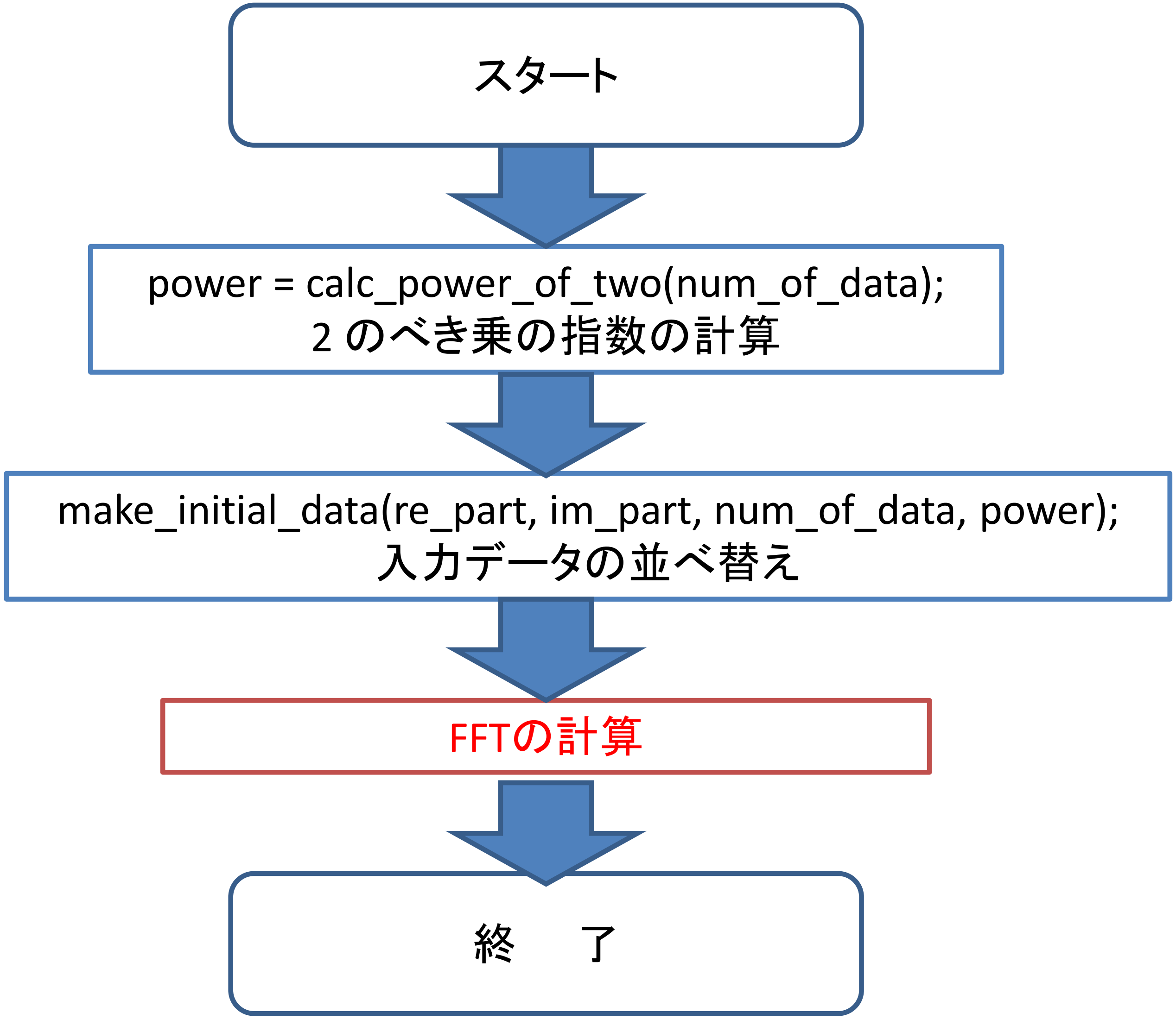


re_part

re_buf

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 4 | 2 | 6 | 1 | 5 | 3 | 7 |

# 全体の流れ（順変換のみに着目）



スタート

power = calc_power_of_two(num_of_data);
2 のべき乗の指数の計算

make_initial_data(re_part, im_part, num_of_data, power);
入力データの並べ替え

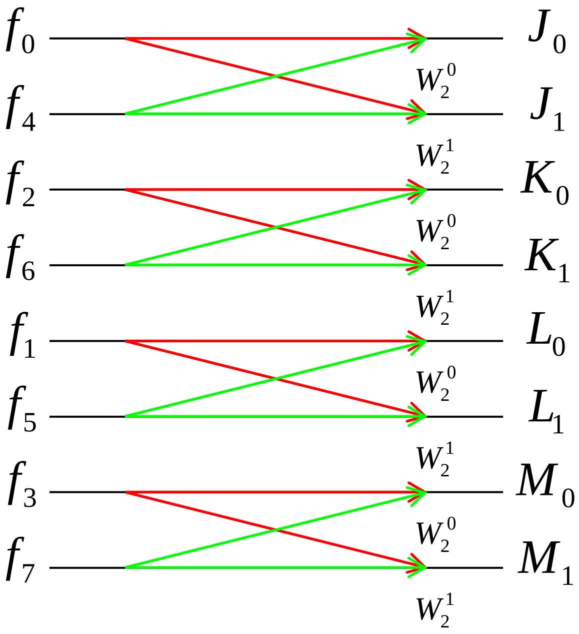
FFTの計算

終　了

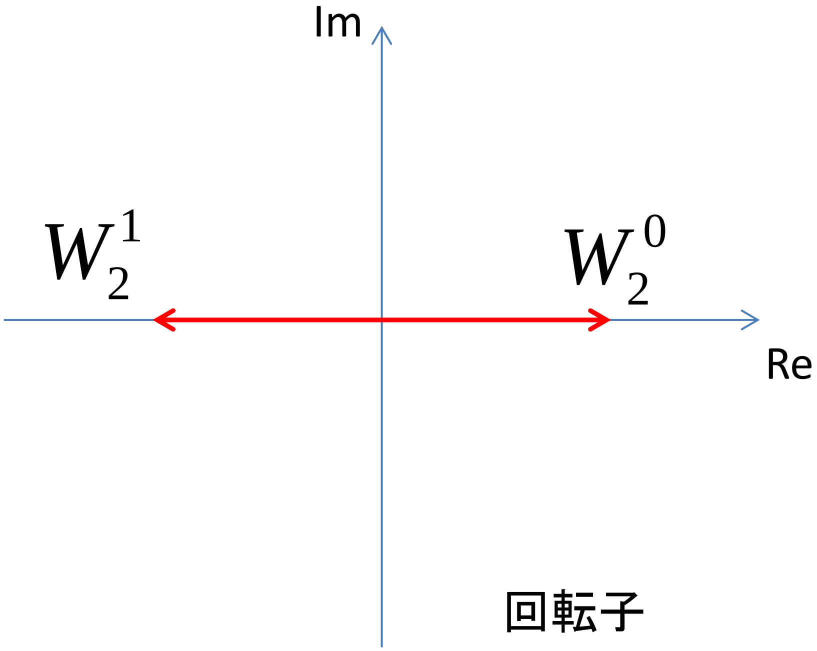
# ステップ1

$$W_N^k = e^{-j\frac{2\pi}{N}k}$$

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

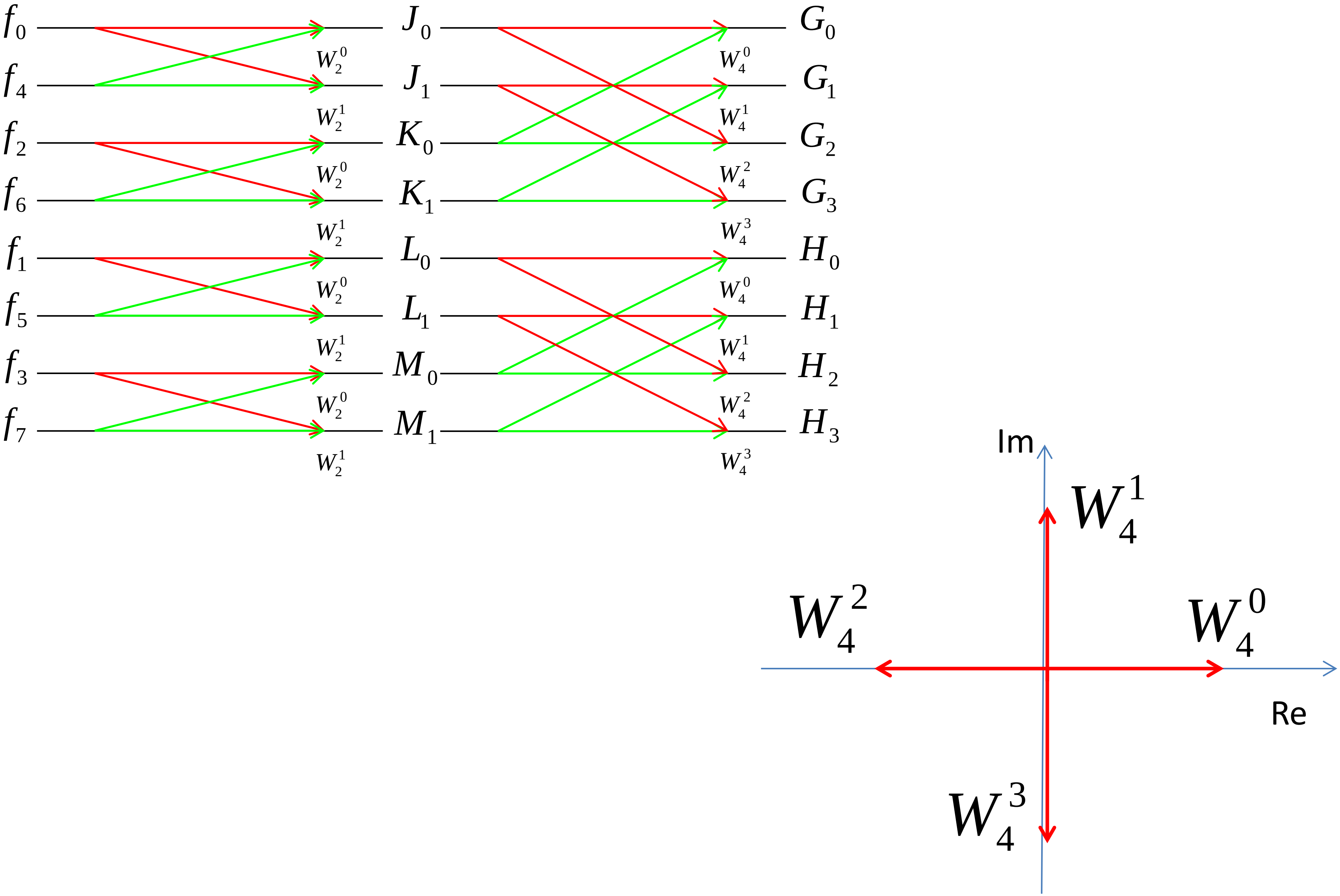


バタフライ



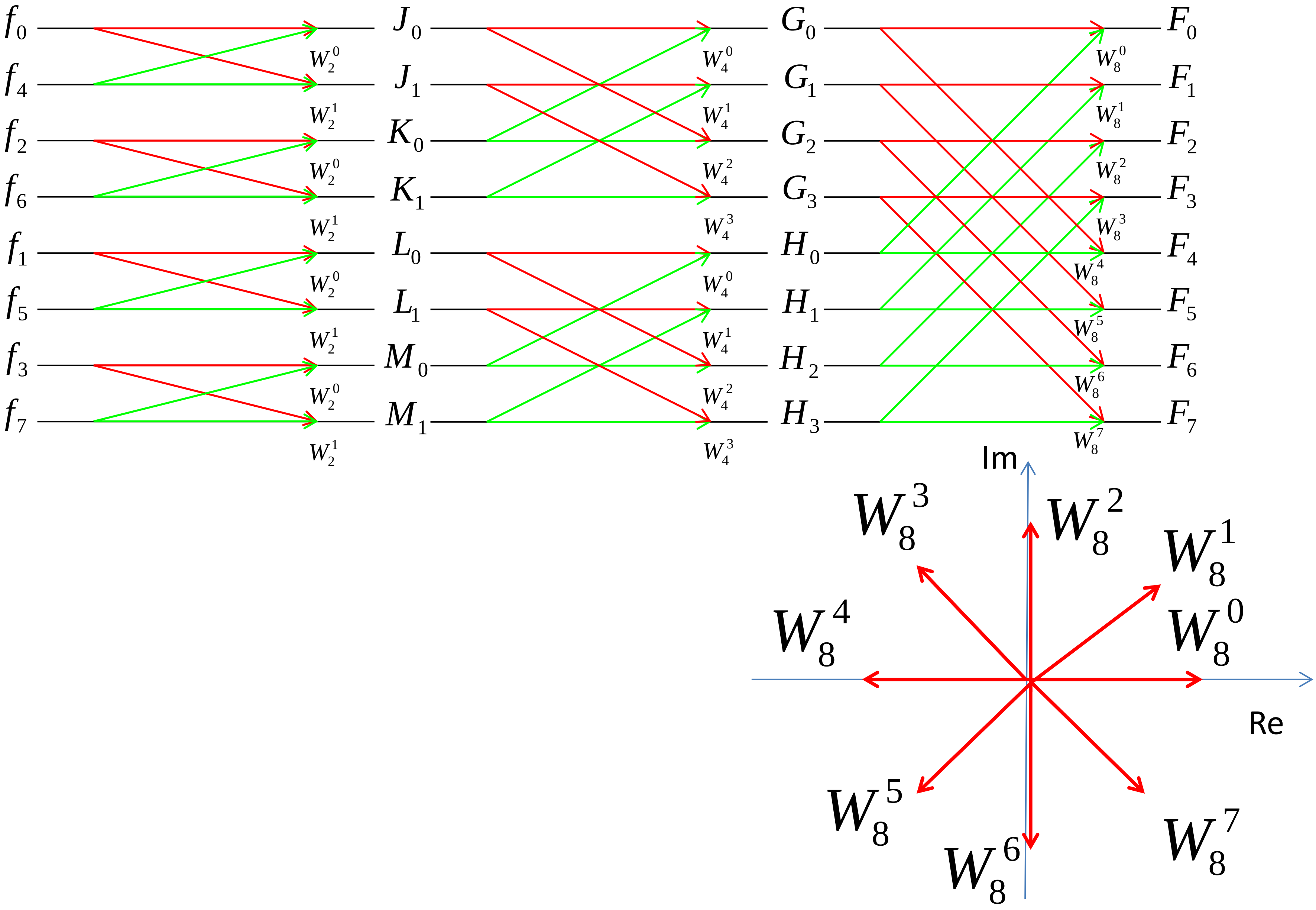
回転子

# ステップ2



$f_0$
$f_4$
$f_2$
$f_6$
$f_1$
$f_5$
$f_3$
$f_7$
$W_2^0$
$W_2^1$
$W_2^0$
$W_2^1$
$W_2^0$
$W_2^1$
$W_2^0$
$W_2^1$
$J_0$
$J_1$
$K_0$
$K_1$
$L_0$
$L_1$
$M_0$
$M_1$
$W_4^0$
$W_4^1$
$W_4^2$
$W_4^3$
$W_4^0$
$W_4^1$
$W_4^2$
$W_4^3$
$G_0$
$G_1$
$G_2$
$G_3$
$H_0$
$H_1$
$H_2$
$H_3$
Im
$W_4^1$
$W_4^2$
$W_4^0$
Re
$W_4^3$

# ステップ3


$f_0$
$f_4$
$f_2$
$f_6$
$f_1$
$f_5$
$f_3$
$f_7$
$W_2^0$
$W_2^1$
$W_2^0$
$W_2^1$
$W_2^0$
$W_2^1$
$W_2^0$
$W_2^1$
$J_0$
$J_1$
$K_0$
$K_1$
$L_0$
$L_1$
$M_0$
$M_1$
$W_4^0$
$W_4^1$
$W_4^2$
$W_4^3$
$W_4^0$
$W_4^1$
$W_4^2$
$W_4^3$
$G_0$
$G_1$
$G_2$
$G_3$
$H_0$
$H_1$
$H_2$
$H_3$
$W_8^0$
$W_8^1$
$W_8^2$
$W_8^3$
$W_8^4$
$W_8^5$
$W_8^6$
$W_8^7$
$F_0$
$F_1$
$F_2$
$F_3$
$F_4$
$F_5$
$F_6$
$F_7$
Im
Re
$W_8^0$
$W_8^1$
$W_8^2$
$W_8^3$
$W_8^4$
$W_8^5$
$W_8^6$
$W_8^7$

```
num_of_dft = num_of_data / dft;
step_angle = unit_angle *  num_of_dft;
half = dft / 2;
for ( j = 0; j < num_of_dft; j ++ ){
    angle = 0.0;
    for ( k = 0; k < dft; k ++ ){
         num_out = j * dft + k;
        if ( k < half ){
          num_in1 = num_out;
          num_in2 = num_in1 + half;
        } else
        {
          num_in2 = num_out;
          num_in1 = num_out - half;
        }
```

```
num_of_dft = 8/2 = 4;
step_angle = (2π/8)*  4 = π;
half = 2/ 2 = 1;
for ( j = 0; j < 4; j ++ ){
    angle = 0.0;
    for ( k = 0; k < 2; k ++ ){
        num_out = 0 * 2+ 0 = 0;
        if ( k < 1){
          num_in1 = 0;
          num_in2 = 0+ 1 = 1;
        } else
        {
          num_in2 = num_out;
          num_in1 = num_out - half;
        }
```

```
/* 実数部(re_)・虚数部(im_)に分けて計算 */
re_buf = *( re_part + num_in2 );
im_buf = *( im_part + num_in2 );
re_part_new[num_out] = *( re_part + num_in1 )
   + re_buf * cos(angle) + im_buf * sin(angle);
im_part_new[num_out] = *( im_part + num_in1 )
   + im_buf * cos(angle) - re_buf * sin(angle);
/* 角度を更新 */
angle = angle + step_angle;
```

```
re_buf = *( re_part  + 1);
im_buf = *( im_part + 1);
re_part_new[0] = *( re_part + 0)
+ re_buf * cos(0) + im_buf * sin(0);
im_part_new[0] = *( im_part + 0)
+ im_buf * cos(0) - re_buf * sin(0);

angle = angle + π = π;
```

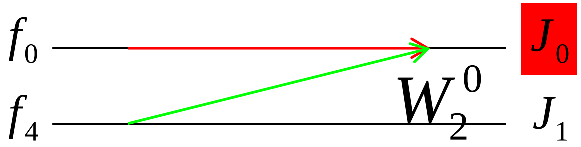
$$\boxed{J_0=f0+((re\_part)+j(im\_part))*(\cos(0)-j\ \sin(0))}$$

$J_0$ : re_part_new =
*(re_part+0)+*(re_part+1)*cos(0)+*(im_part+1)*sin(0)

$J_0$ : im_part_new =
*(im_part+0)+*(im_part+1)*cos(0)-*(re_part+1)*sin(0)

*(re_part+0)

*(im_part+0)



*(re_part+1)

*(im_part+1)

```
num_of_dft = num_of_data / dft;
step_angle = unit_angle *  num_of_dft;
half = dft / 2;
for ( j = 0; j < num_of_dft; j ++ ){
    angle = 0.0;
    for ( k = 0; k < dft; k ++ ){
         num_out = j * dft + k;
        if ( k < half ){
          num_in1 = num_out;
          num_in2 = num_in1 + half;
        } else
        {
          num_in2 = num_out;
          num_in1 = num_out - half;
        }
```

```
num_of_dft = 8/2 = 4;
step_angle = (2π/8)*  4 = π;
half = 2/ 2 = 1;
for ( j = 0; j < 4; j ++ ){
    angle = 0.0;
    for ( k = 0; k < 2; k ++ k=1){
        num_out = 0 * 2+ 1 = 1;
        if ( 1 < 1){
           num_in1 = num_out;
          num_in2 = num_in1 + half;
         } else
        {
          num_in2 = 1;
          num_in1 = 1- 1=0;
        }
```

```
/* 実数部(re_)・虚数部(im_)に分けて計算 */
re_buf = *( re_part + num_in2 );
im_buf = *( im_part + num_in2 );
re_part_new[num_out] = *( re_part + num_in1 )
   + re_buf * cos(angle) + im_buf * sin(angle);
im_part_new[num_out] = *( im_part + num_in1 )
   + im_buf * cos(angle) - re_buf * sin(angle);
/* 角度を更新 */
angle = angle + step_angle;
```

```
re_buf = *( re_part  + 0);
im_buf = *( im_part + 0);
re_part_new[1] = *( re_part + 1)
+ re_buf * cos(π) + im_buf * sin(π);
im_part_new[1] = *( im_part + 1)
+ im_buf * cos(π) - re_buf * sin(π);

angle = angle + π = 2π;
```

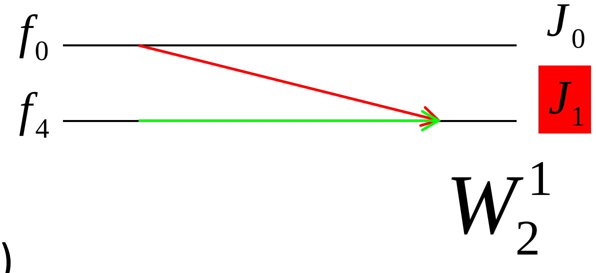
$$\boxed{J_1=f4+((re\_part)+j(im\_part))*(\cos(0)-j\sin(0))}$$

$J_1$ : re_part_new =
*(re_part+1)+*(re_part+0)*cos(π)+*(im_part+0)*sin(π)

$J_1$ : im_part_new =
*(im_part+0)+*(im_part+0)*cos(π)-*(re_part+0)*sin(π)

*(re_part+0)

*(im_part+0)



*(re_part+1)

*(im_part+1)

```
num_of_dft = num_of_data / dft;
step_angle = unit_angle *  num_of_dft;
half = dft / 2;
for ( j = 0; j < num_of_dft; j ++ ){
    angle = 0.0;
    for ( k = 0; k < dft; k ++ ){
         num_out = j * dft + k;
        if ( k < half ){
          num_in1 = num_out;
          num_in2 = num_in1 + half;
        } else
        {
          num_in2 = num_out;
          num_in1 = num_out - half;
        }
```

```
num_of_dft = 8/4 = 2;
step_angle = (2π/8)*  2 = π/2;
half = 4/ 2 = 2;
for ( j = 0; j < 2; j ++ ){
    angle = 0.0;
    for ( k = 0; k < 4; k ++ k=0){
        num_out = 0 * 4+ 0 = 0;
        if ( 0 < 2){
           num_in1 = 0;
          num_in2 =0+ 2=2;
         } else
        {
          num_in2 = num_out;
          num_in1 = num_out - half;
        }
```

```
/* 実数部(re_)・虚数部(im_)に分けて計算 */
re_buf = *( re_part + num_in2 );
im_buf = *( im_part + num_in2 );
re_part_new[num_out] = *( re_part + num_in1 )
   + re_buf * cos(angle) + im_buf * sin(angle);
im_part_new[num_out] = *( im_part + num_in1 )
   + im_buf * cos(angle) - re_buf * sin(angle);
/* 角度を更新 */
angle = angle + step_angle;
```

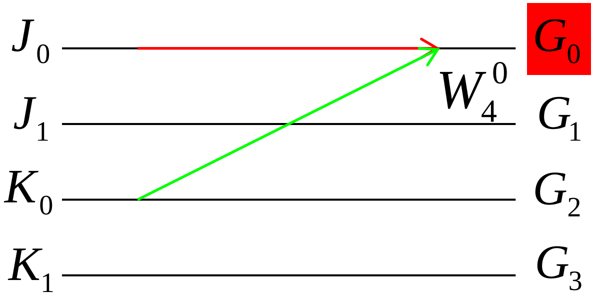
```
re_buf = *( re_part  + 2);
im_buf = *( im_part + 2);
re_part_new[0] = *( re_part + 0)
+ re_buf * cos(0) + im_buf * sin(0);
im_part_new[0] = *( im_part + 0)
+ im_buf * cos(0) - re_buf * sin(0);

angle = angle + π /2= π/2;
```

*(re_part+0)

*(im_part+0)

*(re_part+2)

*(im_part+2)

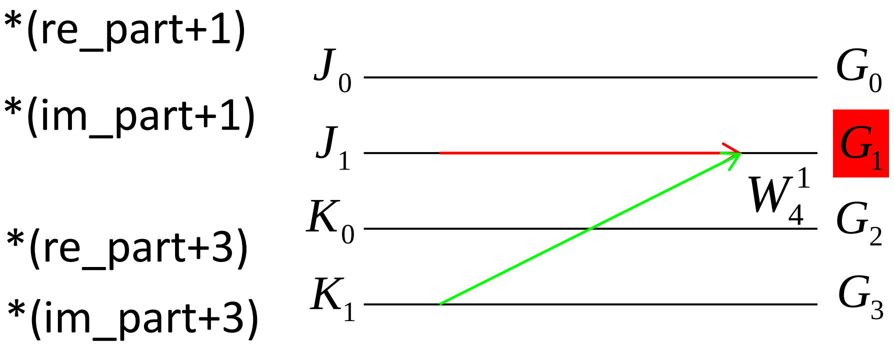


```
num_of_dft = num_of_data / dft;
step_angle = unit_angle *  num_of_dft;
half = dft / 2;
for ( j = 0; j < num_of_dft; j ++ ){
    angle = 0.0;
    for ( k = 0; k < dft; k ++ ){
         num_out = j * dft + k;
        if ( k < half ){
           num_in1 = num_out;
           num_in2 = num_in1 + half;
        } else
        {
           num_in2 = num_out;
           num_in1 = num_out - half;
        }
```

```
num_of_dft = 8/4 = 2;
step_angle = (2π/8)*  2 = π/2;
half = 4/ 2 = 2;
for ( j = 0; j < 2; j ++ ){
    angle = 0.0;
    for ( k = 0; k < 4; k ++ k=1){
        num_out = 0 * 4+ 1 = 1;
        if ( 1 < 2){
           num_in1 = 1;
          num_in2 =1+ 2=3;
         } else
        {
          num_in2 = num_out;
          num_in1 = num_out - half;
        }
```

```
/* 実数部(re_)・虚数部(im_)に分けて計算 */
re_buf = *( re_part + num_in2 );
im_buf = *( im_part + num_in2 );
re_part_new[num_out] = *( re_part + num_in1 )
   + re_buf * cos(angle) + im_buf * sin(angle);
im_part_new[num_out] = *( im_part + num_in1 )
   + im_buf * cos(angle) - re_buf * sin(angle);
/* 角度を更新 */
angle = angle + step_angle;
```

```
re_buf = *( re_part  + 3);
im_buf = *( im_part + 3);
re_part_new[1] = *( re_part + 1)
+ re_buf * cos(π/2) + im_buf * sin(π/2);
im_part_new[1] = *( im_part + 1)
+ im_buf * cos(π/2) - re_buf * sin(π/2);

angle = angle + π /2= π;
```

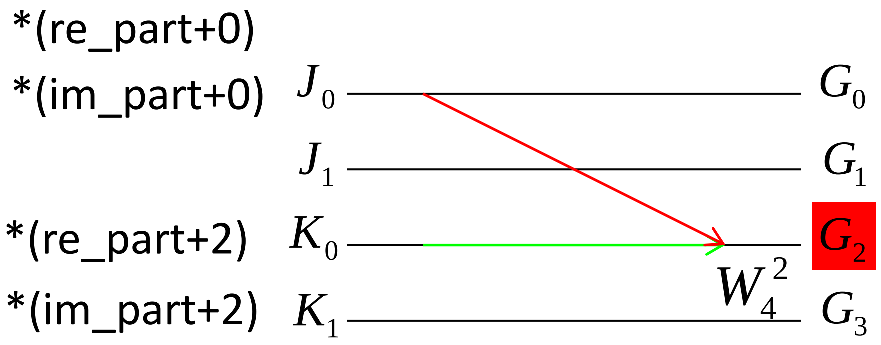


```
num_of_dft = num_of_data / dft;
step_angle = unit_angle *  num_of_dft;
half = dft / 2;
for ( j = 0; j < num_of_dft; j ++ ){
    angle = 0.0;
    for ( k = 0; k < dft; k ++ ){
         num_out = j * dft + k;
        if ( k < half ){
          num_in1 = num_out;
          num_in2 = num_in1 + half;
        } else
        {
          num_in2 = num_out;
          num_in1 = num_out - half;
        }
```

```
num_of_dft = 8/4 = 2;
step_angle = (2π/8)*  2 = π/2;
half = 4/ 2 = 2;
for ( j = 0; j < 2; j ++ ){
    angle = 0.0;
    for ( k = 0; k < 4; k ++ k=2){
        num_out = 0 * 4+ 2 = 2;
        if ( 2< 2){
          num_in1 = num_out;
          num_in2 = num_in1 + half;
         } else
        {
          num_in2 = 2;
          num_in1 = 2- 2=0;
        }
```

```
/* 実数部(re_)・虚数部(im_)に分けて計算 */
re_buf = *( re_part + num_in2 );
im_buf = *( im_part + num_in2 );
re_part_new[num_out] = *( re_part + num_in1 )
   + re_buf * cos(angle) + im_buf * sin(angle);
im_part_new[num_out] = *( im_part + num_in1 )
   + im_buf * cos(angle) - re_buf * sin(angle);
/* 角度を更新 */
angle = angle + step_angle;
```

```
re_buf = *( re_part  + 2);
im_buf = *( im_part + 2);
re_part_new[2] = *( re_part + 0)
+ re_buf * cos(π) + im_buf * sin(π);
im_part_new[2] = *( im_part + 0)
+ im_buf * cos(π) - re_buf * sin(π);

angle = angle + π /2= π;
```



# 課題1

- 関数FFT1を利用して、１次元の時系列信号をフーリエ変換し、スペクトル表示せよ
  - １次元の時系列信号は、複数の異なる周波数成分を持つ正弦波の和とする
  - スペクトルのグラフは表計算ソフトで表示して良い