Fast Fourier Transform

Sogang ICPC Team
2020 Spring 임지환

Contents

Introduction

- 다항식

$$A(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$$

- 다항식 표기법

1) Coefficient Vector

$$< a_0, a_1, a_2, \ldots, a_{n-1} >$$

2) equation by Roots

$$A(x) = c(x - r_0)(x - r_1) \ldots (x - r_{n-1})$$

- Evaluation

$$A(x_k) = a_0 + a_1 x_k + a_2 x_k^2 + \cdots + a_{n-1} x_k^{n-1}$$

- Addition

$$A(x) + B(x) = (a_0 + b_0) + (a_1 + b_1)x + \cdots + (a_{n-1} + b_{n-1})x^{n-1}$$

- Multiplication

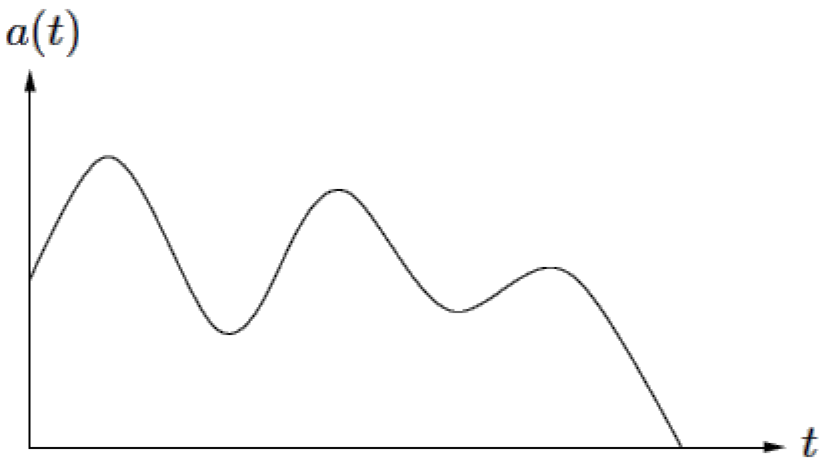
$$A(x) \times B(x) = (a_0 b_0) + (a_0 b_1 + a_1 b_0)x + (a_0 b_2 + a_1 b_1 + a_2 b_0)x^2 + \cdots$$

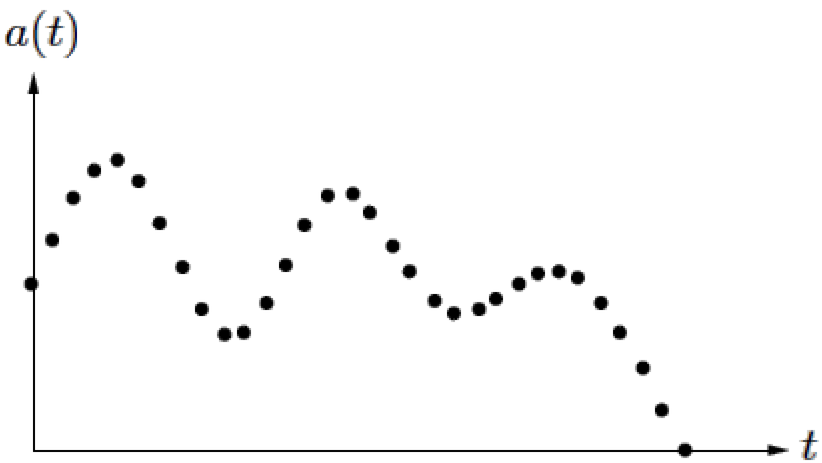
$$= \sum_{k=0}^{n-1} \left\{ \left(\sum_{j=0}^{k} a_j b_{k-j} \right) x^k \right\}$$

Fact: n차 다항식은 서로 다른 n+1개의 점들에 의해 결정될 수 있다.

3) Samples

$$(x_k, y_k) \text{ for } A(x_k) = y_k \ \ (k = 0, 1, \ldots, n-1, \forall x_k\ distinct)$$





	Coefficient vector	equation by Roots	Samples
Evaluation	$O(n)$	$O(n)$	$O(n^2)$
Addition	$O(n)$	∞	$O(n)$
Multiplication	$O(n^2)$	$O(n)$	$O(n)$

Fast Fourier Transform

- Coefficient vector to Sampling by matrix reformulation

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ \vdots \\ A(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ & & \vdots & & \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ & & \vdots & & \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{bmatrix}$$

- $V_{jk} = x_j^k$
- 서로 다른 x_i들에 대하여 역행렬 V^{-1} 존재
- Naïve solution : $O(N^2)$

- goal : get $A(x)$ for $x \in X$ (X: set of x_i)

1. Divide

into even / odd coefficients

$$A_{even}(x) = \sum_{k=0}^{\frac{N}{2}-1} a_{2k}x^k = < a_0, a_2, \dots, a_{n-2} >$$

$$A_{odd}(x) = \sum_{k=0}^{\frac{N}{2}-1} a_{2k+1}x^k = < a_1, a_3, \dots, a_{n-1} >$$

- goal : get $A(x)$ for $x \in X$ (X: set of x_i)

1. Divide

$$A_{even}(x) = \sum_{k=0}^{\frac{N}{2}-1} a_{2k}x^k = < a_0, a_2, \ldots, a_{n-2} >, \qquad A_{odd}(x) = \sum_{k=0}^{\frac{N}{2}-1} a_{2k+1}x^k = < a_1, a_3, \ldots, a_{n-1} >$$

2. Conquer

=> recursively compute

$A_{even}(x^2)$ & $A_{odd}(x^2)$ for $x^2 \in X^2$ $(X^2 = \{x^2 | x \in X\})$

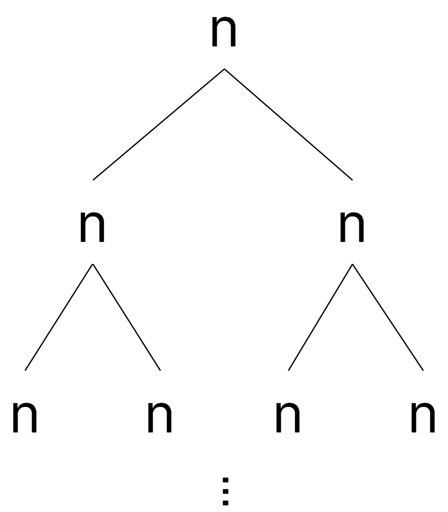
- goal : get $A(x)$ for $x \in X$ (X: set of x_i)

3. Combine

$$A(x) = A_{even}(x^2) + x\,A_{odd}(x^2)$$

- Time Complexity Analysis

$$T(n, |x|) = 2T\left(\frac{n}{2}, |X^2|\right) + O(n + |X^2|)$$



$$O(n^2)$$

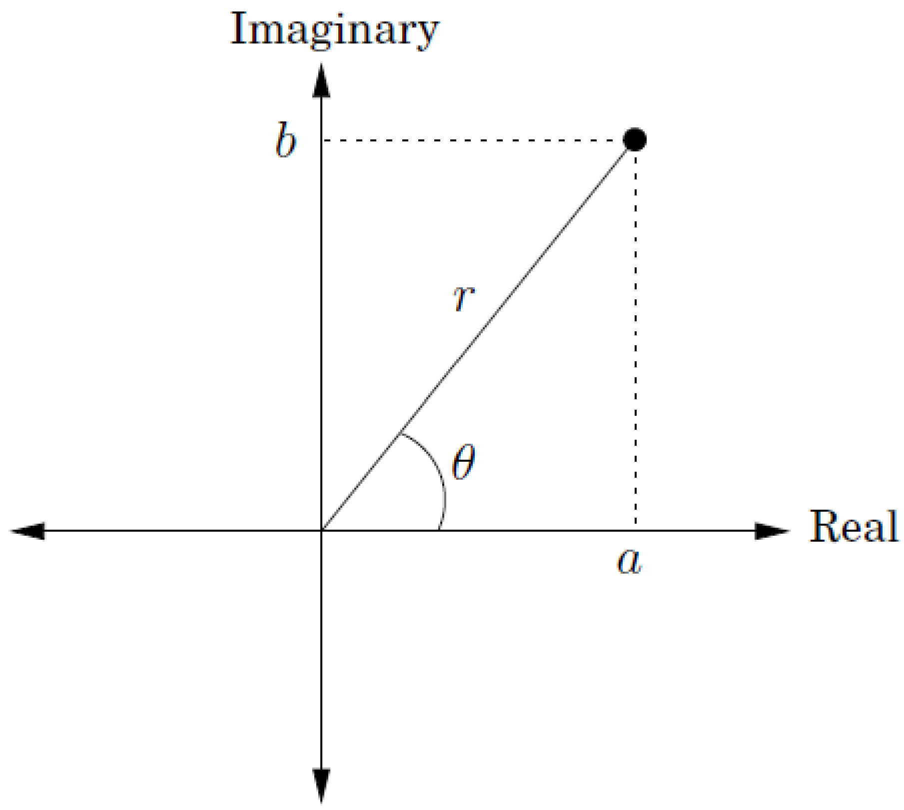
- TODO : half the size of $|X^2|$ $\text{s.t. } x^2 \in X^2$

- half the size of $|X^2|$ s.t. $x^2 \in X^2$

$$|X| = 1 : \ X = \{ 1 \}$$
$$|X| = 2 : \ X = \{ \pm 1 \}$$
$$|X| = 4 : \ X = \{ \pm i, \ \pm 1 \}$$
$$|X| = 8 : \ X = \{ \pm \frac{\sqrt{2}}{2}(1+i), \pm \frac{\sqrt{2}}{2}(i-1), \pm i, \pm 1 \}$$

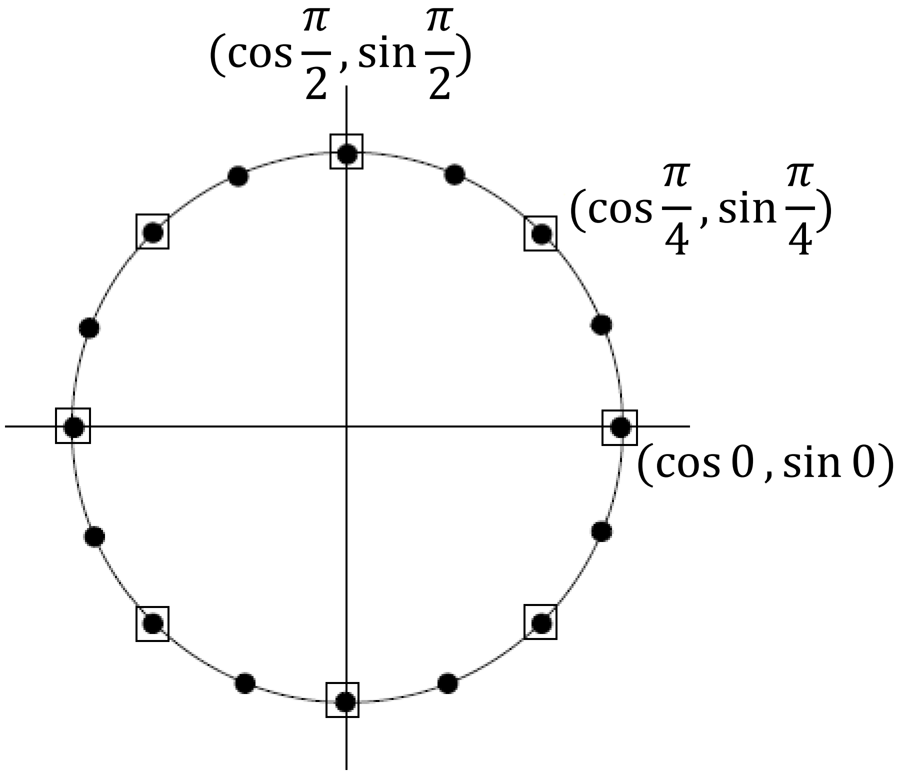
- Collapsing set
  1) $|X| = 1$
  2) a set that $|X^2| = \frac{|X|}{2}$ and $|X^2|$ is collapsing

- $z = a + bi = (a,\ b)$



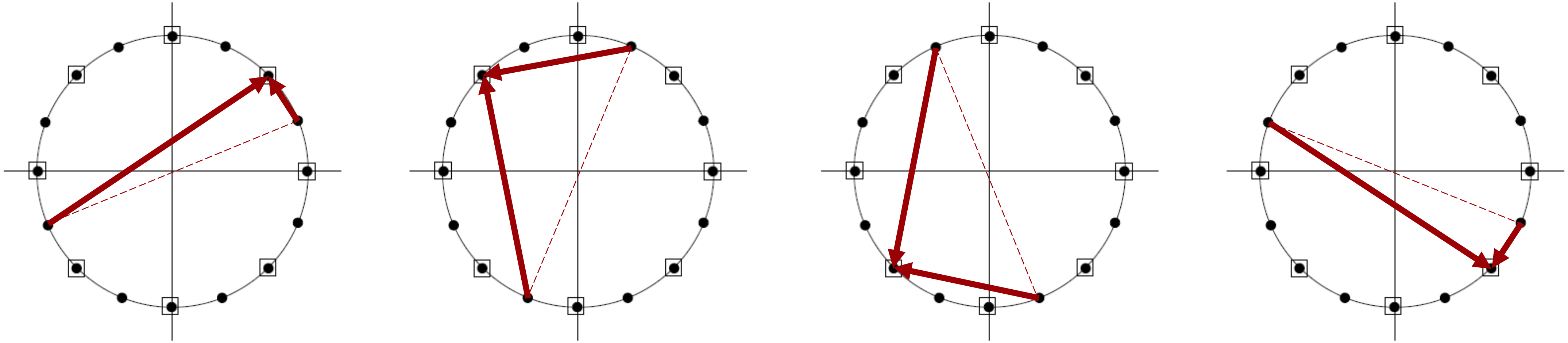
$$1 = \cos(0) + i\sin(0)$$
$$i = \cos\frac{\pi}{2} + i\sin\frac{\pi}{2}$$
$$\frac{\sqrt{2}}{2}(1 + i) = \cos\frac{\pi}{4} + i\sin\frac{\pi}{4}$$

- $\theta = 0, \frac{2\pi}{n}, \frac{2}{n} 2\pi, \ldots, \frac{n-1}{n} 2\pi$
- $z = \cos\theta + i\sin\theta = e^{i\theta}$ (by Euler's formula)



- nth root of unity: $x_k = e^{i\frac{k}{n}2\pi} \quad (k = 0, 1, 2, \dots, n-1)$
- $\left(e^{i\theta}\right)^2 = e^{i(2\theta)}$

squaring == doubling angle == halving the size of $|X^2|$



- goal : get $A(x)$ for $x \in X$ (X: set of x_i)

3. Combine

$$A(x) = A_{even}(x^2) + x\,A_{odd}(x^2)$$

- Coefficient vector to Sampling

$\Rightarrow A^* = V \cdot A$ for $x_k = e^{ik\frac{2\pi}{n}}$ where $V_{jk} = x_j^k = e^{i\left(jk\frac{2\pi}{n}\right)}$

The fast Fourier transform (polynomial formulation)

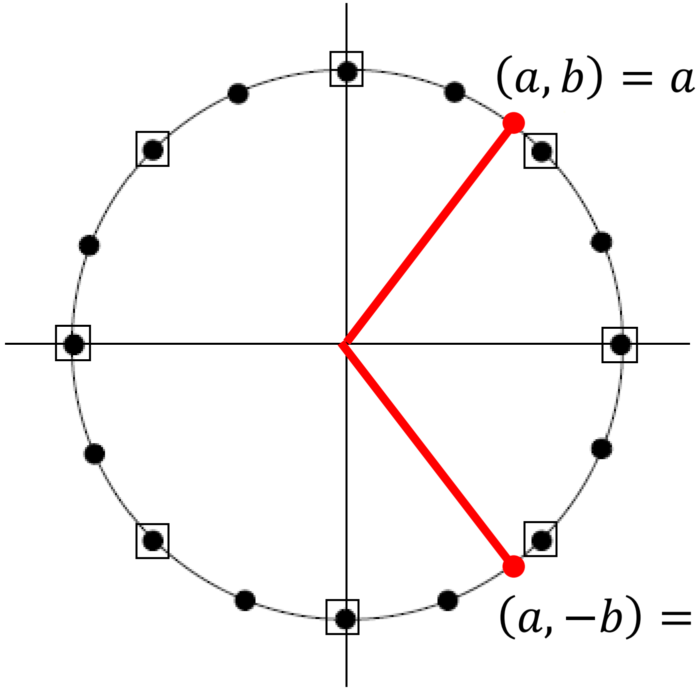
```
function FFT(A, ω)
Input:  Coefficient representation of a polynomial A(x)
        of degree ≤ n−1, where n is a power of 2
        ω, an nth root of unity
Output:  Value representation A(ω^0), ..., A(ω^(n−1))

if ω = 1:  return A(1)
express A(x) in the form A_e(x^2) + x A_o(x^2)
call FFT(A_e, ω^2) to evaluate A_e at even powers of ω
call FFT(A_o, ω^2) to evaluate A_o at even powers of ω
for j = 0 to n−1:
   compute A(ω^j) = A_e(ω^(2j)) + ω^j A_o(ω^(2j))

return A(ω^0), ..., A(ω^(n−1))
```

- Sampling to Coefficient Vector

$\Rightarrow A = V^{-1} \cdot A^*$

claim: $V^{-1} = \frac{\bar{V}}{n}$ where $\overline{a + ib} = a - ib$

then $nA = \bar{V} \cdot A^*$



pf) Let $P = V \cdot \bar{V}$, then $P = nI$ 임을 보이자.

$$P_{jk} = (row\, j\, of\, V) \cdot (col\, k\, of\, \bar{V}) = \sum_{m=0}^{n-1} x_j^m \cdot x_m^k$$

$$= \sum_{m=0}^{n-1} \left(e^{ij\frac{2\pi}{n}}\right)^m \cdot \left(e^{im\frac{2\pi}{n}}\right)^k = \sum_{m=0}^{n-1} e^{im\frac{2\pi}{n}(j-k)}$$

1) $j = k$, then $e^{im\frac{2\pi}{n}(j-k)} = e^{i\cdot 0} = 1$, thus $P_{jk} = n$

2) $j \neq k$, then

$$\sum_{m=0}^{n-1} \left(e^{i\,2\pi\frac{j-k}{n}}\right)^m = \frac{\left(e^{i\,2\pi\frac{j-k}{n}}\right)^n - 1}{\left(e^{i\,2\pi\frac{j-k}{n}}\right) - 1} = \frac{e^{i\,2\pi(j-k)} - 1}{\left(e^{i\,2\pi\frac{j-k}{n}}\right) - 1} = 0$$

$(\because e^{i\,2\pi} = 1)$

- myungwoo's: https://blog.myungwoo.kr/54
- shiftpsh's: https://gist.github.com/shiftpsh/5f1a636995cf4ebbef3633b8800fcc6c

- 수의 길이 : 300,000 이하

					a_2	a_1	a_0
$\times$					b_2	b_1	b_0
			a_2b_2	$a_1b_2 + a_2b_1$	$a_0b_2 + a_1b_1 + a_2b_0$	$a_0b_1 + a_1b_0$	a_0b_0

- 수의 길이 : 300,000 이하

$a_7(=0)$	$a_6(=0)$	$a_5(=0)$	$a_4(=0)$	$a_3(=0)$	a_2	a_1	a_0

$\times$

$b_7(=0)$	$b_6(=0)$	$b_5(=0)$	$b_4(=0)$	$b_3(=0)$	b_2	b_1	b_0

$\sum_{j=0}^{7} a_j b_{k-j}$	$\sum_{j=0}^{6} a_j b_{k-j}$	$\sum_{j=0}^{5} a_j b_{k-j}$	$\sum_{j=0}^{4} a_j b_{k-j}$	$\sum_{j=0}^{3} a_j b_{k-j}$	$\sum_{j=0}^{2} a_j b_{k-j}$	$\sum_{j=0}^{1} a_j b_{k-j}$	$\sum_{j=0}^{0} a_j b_{k-j}$

1. 작은 자리수부터 계산을 하여 carry를 넘겨주어야 하니 주어진 배열 뒤집기

2. max(두 수의 자리수)보다 큰 2의 거듭제곱만큼 배열 확장

```
	int n = 1;
	while(n < max(sz(a), sz(b))) n <<= 1;
	n<<=2;
	fa.resize(n); fb.resize(n);
```

3. 두 수의 convolution 구하기

```
	fft(fa, false); fft(fb, false);
	for(int i=0; i<n; i++) fa[i] *= fb[i];
	fft(fa, true);
	res.resize(n);
	for(int i=0; i<n; i++) res[i] = int(fa[i].real()+(fa[i].real()>0?0.5:-0.5));
```

- 준소수(semiprime): 두 소수의 곱
- 르모앙의 추측: 5보다 큰 홀수는 홀수 소수 하나와 짝수 준소수 하나의 합으로 나타낼 수 있다.

- $1 \le tc \le 10^5$
- 홀수 $N(5 < N \le 10^6)$이 주어졌을 때, 홀수 소수 하나와 짝수 준소수 하나의 합으로 나타내는 방법의 수를 구하여라.

- 두 소수 $p, q(p \neq 2)$에 대하여 $N = p + 2q$인 경우의 수

	1	2	3	4	5	6	7	8	...
소수$(p \neq 2)$	X	X	O	X	O	X	O	X	...
짝수 준소수	X	X	X	O	X	O	X	X	...

$$0x^0 + 0 \cdot x^1 + 0 \cdot x^2 + 1 \cdot x^3 + 0 \cdot x^4 + 1 \cdot x^5 + 0 \cdot x^6 + 1 \cdot x^7 + \cdots$$
$$0x^0 + 0 \cdot x^1 + 0 \cdot x^2 + 0 \cdot x^3 + 1 \cdot x^4 + 0 \cdot x^5 + 1 \cdot x^6 + 0 \cdot x^7 + \cdots$$

- 주어진 N에 대하여 x^N의 계수인 c_N을 fft를 통해 구할 수 있다.

```
const int mxn = (1<<20)+1;
bitset<mxn> isp;
vector<int> a(mxn), plist(mxn), res(mxn);

void sieve() {
    isp.set();
    isp[0] = isp[1] = 0;
    for(int i = 2; i< mxn; i++) {
        if(!isp[i]) continue;
        if(i*2 < mxn) a[i*2] = 1;
        if(i%2 && i< mxn) plist[i] = 1;
        for(int j = i*2; j< mxn; j+= i)
            isp[j] = 0;
    }
}
```

- 두 polynomial의 곱셈을 $O(N \log N)$에 수행

- 신호처리 등에 사용

- https://solved.ac/problems/algorithms/28

자료 출처

1. MIT 6.046 Design and Analysis of Algorithms, Spring 2015
   https://www.youtube.com/watch?v=iTMn0Kt18tg
2. Sanjoy Dasgupta Christos Papadimitriou, - Algorithms (McGraw-Hill), p68-82