05. Matrix

2019 Summer / 20141574 임지환

Matrix

- 수들을 2차원 배열 형태로 표현한 것.

$$\begin{bmatrix} 6 & 13 & 7 & 4 \\ 7 & 0 & 8 & 2 \\ 9 & 5 & 4 & 18 \end{bmatrix}$$

Matrix

- 수들을 2차원 배열 형태로 표현한 것.

Row vector

$$\begin{bmatrix} \boxed{6 & 13 & 7 & 4} \\ 7 & 0 & 8 & 2 \\ 9 & 5 & 4 & 18 \end{bmatrix}$$

Matrix

- 수들을 2차원 배열 형태로 표현한 것.

$$\begin{bmatrix} \boxed{\begin{matrix} 6 \\ 7 \\ 9 \end{matrix}} & \begin{matrix} 13 \\ 0 \\ 5 \end{matrix} & \begin{matrix} 7 \\ 8 \\ 4 \end{matrix} & \begin{matrix} 4 \\ 2 \\ 18 \end{matrix} \end{bmatrix}$$

Row vector

Notations

- $M \times N\ matrix\ A : A_{mn}$
- $idenety\ matrix\ I$: 정사각행렬 A에서 대각 성분만 1로 구성.

Basic Matrix operations

- 행렬의 합

  행의 개수와 열의 개수가 같을 때 정의

$$\begin{bmatrix} 6 & 13 & 7 & 4 \\ 7 & 0 & 8 & 2 \\ 9 & 5 & 4 & 18 \end{bmatrix} + \begin{bmatrix} 2 & 1 & 9 & -10 \\ -3 & 0 & 3 & 8 \\ 7 & 4 & 16 & 13 \end{bmatrix} = \begin{bmatrix} 8 & 14 & 16 & -6 \\ 4 & 0 & 11 & 10 \\ 16 & 9 & 20 & 31 \end{bmatrix}$$

Basic Matrix operations

- 행렬의 곱

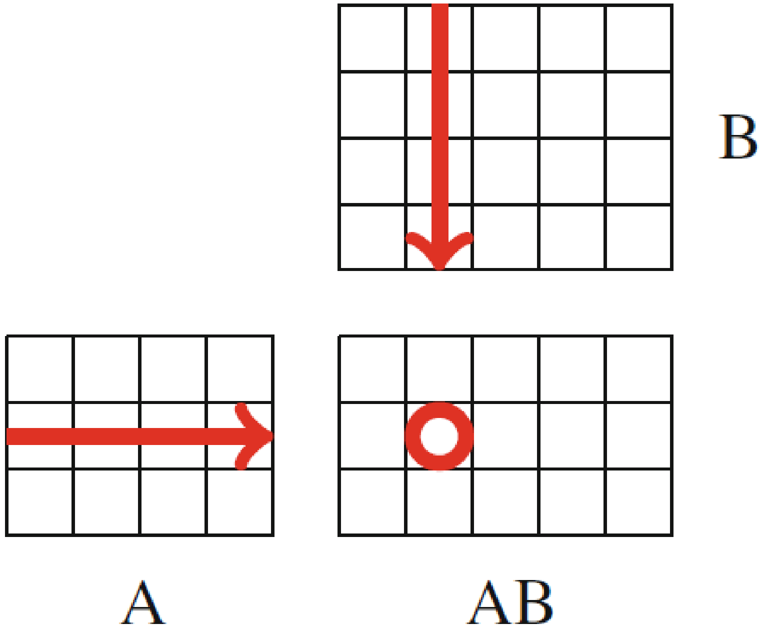
  Multiplying matrix A by a value x

$$2 \cdot \begin{bmatrix} 6 & 1 & 4 \\ 3 & 9 & 2 \end{bmatrix} = \begin{bmatrix} 12 & 2 & 8 \\ 6 & 18 & 4 \end{bmatrix}$$

Basic Matrix operations

- 행렬의 곱

  Multiplying matrix A matrix B



Basic Matrix operations

- 행렬의 곱

  Multiplying matrix A matrix B

$$\begin{bmatrix} 1 & 4 \\ 3 & 9 \\ 8 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 & 6 \\ 2 & 9 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 4 \cdot 2 & 1 \cdot 6 + 4 \cdot 9 \\ 3 \cdot 1 + 9 \cdot 2 & 3 \cdot 6 + 9 \cdot 9 \\ 8 \cdot 1 + 6 \cdot 2 & 8 \cdot 6 + 6 \cdot 9 \end{bmatrix} = \begin{bmatrix} 9 & 42 \\ 21 & 99 \\ 20 & 102 \end{bmatrix}$$

Basic Matrix operations

- 행렬의 곱

  Multiplying matrix A_{nn} matrix B_{nn}

```
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        for (int k = 0; k < n; k++)
            C[i][j] += A[i][k] * B[k][j];
```

https://www.acmicpc.net/problem/2740
faster algorithm : https://en.wikipedia.org/wiki/Strassen_algorithm

Properties of Matrix Multiplication

- **Associativity**

$$A(BC) = (AB)C$$

- ~~**Commutativity**~~

$$AB \neq BA$$

#11049 행렬 곱셈 순서 3

크기가 N×M인 행렬 A와 M×K인 B를 곱할 때 필요한 곱셈 연산의 수는 총 N×M×K번이다. 행렬 N개를 곱하는데 필요한 곱셈 연산의 수는 행렬을 곱하는 순서에 따라 달라지게 된다.

예를 들어, A의 크기가 5×3이고, B의 크기가 3×2, C의 크기가 2×6인 경우에 행렬의 곱 ABC를 구하는 경우를 생각해보자.

- AB를 먼저 곱하고 C를 곱하는 경우 (AB)C에 필요한 곱셈 연산의 수는 5×3×2 + 5×2×6 = 30 + 60 = 90번이다.
- BC를 먼저 곱하고 A를 곱하는 경우 A(BC)에 필요한 곱셈 연산의 수는 3×2×6 + 5×3×6 = 36 + 90 = 126번이다.

같은 곱셈이지만, 곱셈을 하는 순서에 따라서 곱셈 연산의 수가 달라진다.

행렬 N개의 크기가 주어졌을 때, 모든 행렬을 곱하는데 필요한 곱셈 연산 횟수의 최솟값을 구하는 프로그램을 작성하시오. 입력으로 주어진 행렬의 순서를 바꾸면 안 된다.

입력

첫째 줄에 행렬의 개수 N(1 ≤ N ≤ 500)이 주어진다.

둘째 줄부터 N개 줄에는 행렬의 크기 r과 c가 주어진다. (1 ≤ r, c ≤ 500)

항상 순서대로 곱셈을 할 수 있는 크기만 입력으로 주어진다.

#11049 행렬 곱셈 순서 3

- A_{NM}, B_{MK}를 곱할 때 필요한 곱셉 연산의 수 : $N \times M \times K$
- 행렬 곱셈연산 한번 시행 시 행렬의 개수 : $N \rightarrow N-1$
- 마지막 단계에서의 행렬의 개수 : 2

#11049 행렬 곱셈 순서 3

$$A_1 \times A_2 \times A_3 \times \cdots \times A_k \times A_{k+1} \times \cdots \times A_{n-1} \times A_n$$

#11049 행렬 곱셈 순서 3

$$\overbrace{A_1 \times A_2 \times A_3 \times \cdots \times A_k}^{X} \times \overbrace{A_{k+1} \times \cdots \times A_{n-1} \times A_n}^{Y}$$

$X \times Y$가 optimal solution이라면?

#11049 행렬 곱셈 순서

$$\overbrace{A_1 \times A_2 \times A_3 \times \cdots \times A_k}^{X} \times \overbrace{A_{k+1} \times \cdots \times A_{n-1} \times A_n}^{Y}$$

1. 구간 $1 \sim N$에서 $X \times Y$가 최소가 되는 k 찾기

$$solve(1, N) = \min_{i<k<j}(solve(1, k) + solve(k+1, N) + A_1.row \cdot A_k.col \cdot A_n.col$$

#11049 행렬 곱셈 순서 3

$$\overbrace{A_1 \times A_2 \times A_3 \times \cdots \times A_k}^{X} \times \overbrace{A_{k+1} \times \cdots \times A_{n-1} \times A_n}^{Y}$$

1. 구간 1~N에서 $X \times Y$가 최소가 되는 k 찾기

$$solve(1, N) = \min_{1<k<N} (solve(1, k) + solve(k+1, N) + A_1.row \cdot A_k.col \cdot A_n.col$$

$$= X.row \cdot X.col \cdot Y.col$$

#11049 행렬 곱셈 순서 3

$$\overbrace{A_1 \times A_2 \times A_3 \times \cdots \times A_k}^{X} \times \overbrace{A_{k+1} \times \cdots \times A_{n-1} \times A_n}^{Y}$$

2. 점화식 일반화

$$solve(i,j) = \min_{i<k<j}(solve(i,k) + solve(k+1,j) + A_i.row \cdot A_k.col \cdot A_j.col$$

Linear systems of Equation

- system of linear equation(연립일차방정식)

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\
\vdots \\
a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n &= b_m
\end{aligned}
$$

Linear systems of Equation

- system of linear equation(연립일차방정식)

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\
\vdots \\
a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n &= b_m
\end{aligned}
\quad \Rightarrow \quad
\begin{pmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\
a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\
a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\
\vdots & \vdots & \vdots & & \vdots \\
a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn}
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}
=
\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix}
$$

Linear systems of Equation

- To solve equations : Gauss Elimination

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix} \Rightarrow \left[\begin{array}{cccc|c} 1 & 0 & \cdots & 0 & c_1 \\ 0 & 1 & \cdots & 0 & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & c_n \end{array}\right]$$

Elementary Row operation

1. 치환

$$\begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1,n+1} \\ \vdots & \vdots & & \vdots \\ M_{i1} & M_{i2} & \cdots & M_{i,n+1} \\ \vdots & \vdots & & \vdots \\ M_{j1} & M_{j2} & \cdots & M_{j,n+1} \\ \vdots & \vdots & & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{m,n+1} \end{pmatrix} x = 0 \implies \begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1,n+1} \\ \vdots & \vdots & & \vdots \\ M_{j1} & M_{j2} & \cdots & M_{j,n+1} \\ \vdots & \vdots & & \vdots \\ M_{i1} & M_{i2} & \cdots & M_{i,n+1} \\ \vdots & \vdots & & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{m,n+1} \end{pmatrix} x = 0$$

Elementary Row operation

2. 상수곱

$$\begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1,n+1} \\ \vdots & \vdots & & \vdots \\ M_{i1} & M_{i2} & \cdots & M_{i,n+1} \\ \vdots & \vdots & & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{m,n+1} \end{pmatrix} x = 0 \Longrightarrow \begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1,n+1} \\ \vdots & \vdots & & \vdots \\ aM_{i1} & aM_{i2} & \cdots & aM_{i,n+1} \\ \vdots & \vdots & & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{m,n+1} \end{pmatrix} x = 0$$

Elementary Row operation

3. 합

$$\begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1,n+1} \\ \vdots & \vdots & & \vdots \\ M_{i1} & M_{i2} & \cdots & M_{i,n+1} \\ \vdots & \vdots & & \vdots \\ M_{j1} & M_{j2} & \cdots & M_{j,n+1} \\ \vdots & \vdots & & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{m,n+1} \end{pmatrix} x = 0 \implies \begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1,n+1} \\ \vdots & \vdots & & \vdots \\ M_{i1} & M_{i2} & \cdots & M_{i,n+1} \\ \vdots & \vdots & & \vdots \\ aM_{i1}+M_{j1} & aM_{i2}+M_{j2} & \cdots & aM_{i,n+1}+M_{j,n+1} \\ \vdots & \vdots & & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{m,n+1} \end{pmatrix} x = 0$$

Row Echelon Form

- 행사다리꼴 행렬

1. 선행 계수(or leading 1)는 항상 그 위 행의 선행계수의 오른쪽에 위치.
2. 모든 zero row vector들보다 위에 위치

$$\begin{bmatrix} 2 & 2 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 4 & -6 & 0 & -2 \\ -2 & 7 & 2 & 9 \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 4 & -6 & 0 & -2 \\ -2 & 7 & 2 & 9 \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 4 & -6 & 0 & -2 \\ -2 & 7 & 2 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 4-(2\times 2) & -6-(2\times 1) & 0-(2\times 1) & -2-(2\times 5) \\ -2 & 7 & 2 & 9 \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ -2 & 7 & 2 & 9 \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ -2 & 7 & 2 & 9 \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ -2 & 7 & 2 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ -2-(-1\times 2) & 7-(-1\times 1) & 2-(-1\times 1) & 9-(-1\times 5) \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ 0 & 8 & 3 & 14 \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ 0 & 8 & 3 & 14 \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ 0 & 8 & 3 & 14 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ 0-(-1\times 0) & 8-(-1\times -8) & 3-(-1\times -2) & 14-(-1\times -12) \end{bmatrix}$$

Row Echelon Form

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & -2 & -12 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

Linear combination

- 벡터들의 집합 $v_1, \dots, v_n \in V$ 와 계수집합 $c_1, \dots, c_n \in K$에 의한 선형결합 u :

$$u = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n = \sum_{i=1}^{n} c_i v_i$$

Linear Independence & Basis

- Linear Independent

  $c_1 v_1 + c_2 v_2 + \cdots + c_n v_n = 0$를 만족하는 $c_1, \ldots, c_n$이 모두 0인 관계를 만족하는 벡터

- Basis

  linear Independent를 만족하는 벡터 집합

  ex) 3차원 좌표계에서의 (1,0,0), (0,1,0), (0,0,1)

#11191 Xor Maximiaztion 4

- $n\,(1 \le n \le 100{,}000)$개의 정수 $a_1, a_2, \ldots, a_n(1 \le a_i \le 10^{18})$가 주어졌을 때, 이들 중 몇 개를 적절히 골라내어 xor연산을 하였을 때의 결과가 최대가 되도록 한 값을 구하여라.

#11191 Xor Maximiaztion 4

- 이진수에서의 XOR

  이진수끼리의 xor연산은 덧셈연산과 동일 (carry를 배제한 경우)

- 수 집합의 XOR 연산의 최대값 = Basis의 linear combination의 최대값

#11191 Xor Maximiaztion 4

$Row[i]$: $leading$ 1의 위치가 i번째 column에 등장하는 basis

```
lint basis[60], x;

void computeREF(int x) {
    for (int i = 59; i >= 0; i--) {
        if ((x >> i) & 1) {
            if (!basis[i]) {
                basis[i] = x;
                return;
            }
            else x ^= basis[i]; // elimination
        }
    }
}
```

more info : https://raararaara.blog.me/221511605448

Linear Recurrences

- Definition

initial value가 $f(0), f(1), \dots, f(k-1)$일 때

$$f(n) = c_1 f(n-1) + c_2 f(n-2) + \cdots + c_k f(n-k)$$

와 같이 정의되는 함수

#11444 피보나치 수 6 4

- $n (\le 10^{18})$번째 피보나치 수를 1,000,000,007로 나눈 나머지를 구하여라.

#11444 피보나치 수 6 4

- 기존에 배운 Dynamic Programming

$$f(n) = f(n-1) + f(n-2)$$

#11444 피보나치 수 6 4

- 기존에 배운 Dynamic Programming

$$f(n) = f(n-1) + f(n-2)$$

$$\Downarrow$$

$$O(n)$$

#11444 피보나치 수 6 4

- 기존에 배운 Dynamic Programming

$$f(n) = f(n-1) + f(n-2)$$

#11444 피보나치 수 6 4

- 기존에 배운 Dynamic Programming

$$f(n) = 1 \cdot f(n-1) + 1 \cdot f(n-2)$$

#11444 피보나치 수 6 4

- 기존에 배운 Dynamic Programming

$$f(n) = 1 \cdot f(n-1) + 1 \cdot f(n-2)$$

$$\Downarrow$$

Linear Recurrence

#11444 피보나치 수 6 4

- Solving Linear Recurrence By Matrix Multiplication

$$X \cdot \begin{bmatrix} f_i \\ f_{i+1} \end{bmatrix} = \begin{bmatrix} f_{i+1} \\ f_{i+2} \end{bmatrix}$$

#11444 피보나치 수 6 4

- Solving Linear Recurrence By Matrix Multiplication

$$put\ X = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} f_i \\ f_{i+1} \end{bmatrix} = \begin{bmatrix} f_{i+1} \\ f_{i+2} \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \begin{bmatrix} f_0 \\ f_1 \end{bmatrix} = \begin{bmatrix} f_n \\ f_{n+1} \end{bmatrix}$$

#11444 피보나치 수 6 4

- 행렬의 n제곱?

#1629 곱셈 3

자연수 A를 B번 곱한 수를 알고 싶다. 단 구하려는 수가 매우 커질 수 있으므로 이를 C로 나눈 나머지를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 A, B, C가 빈 칸을 사이에 두고 순서대로 주어진다. A, B, C는 모두 2,147,483,647 이하의 자연수이다.

#1629 곱셈 3

- A^B의 $O(B)$보다 빠른 연산
  1. B를 이진수로 표현

#1629 곱셈 3

- A^B의 $O(B)$보다 빠른 연산

    1. B를 이진수로 표현(ex; B = 59)

#1629 곱셈 3

- A^B의 $O(B)$보다 빠른 연산
  1. B를 이진수로 표현(ex; B = 59)

     B = 111011_2

#1629 곱셈 3

- A^B의 $O(B)$보다 빠른 연산
  1. B를 이진수로 표현(ex; B = 59)

     B = 111011_2
  2. LSB부터 차례로 연산

#1629 곱셈 3

- A^B의 $O(B)$보다 빠른 연산

1. B를 이진수로 표현(ex; B = 59)

   B = 111011_2

2. LSB부터 차례로 연산

   $A^{59} = A^1 \cdot A^2 \cdot A^8 \cdot A^{16} \cdot A^{32}$

#1629 곱셈 3

- A^B의 $O(B)$보다 빠른 연산
  1. B를 이진수로 표현(ex; B = 59)

     $B = 111011_2$
  2. LSB부터 차례로 연산

     $A^{59} = A^1 \cdot A^2 \cdot A^8 \cdot A^{16} \cdot A^{32}$

```
while (b > 0) {
    if (b % 2) {
        res *= a;
        res %= c;
    }
    a *= a;
    a %= c;
    b = b >> 1;
}
```

#1629 곱셈 3

- A^B의 $O(B)$보다 빠른 연산
  1. B를 이진수로 표현(ex; B = 59)

     $B = 111011_2$
  2. LSB부터 차례로 연산

     $A^{59} = A^1 \cdot A^2 \cdot A^8 \cdot A^{16} \cdot A^{32}$

```
while (b > 0) {
    if (b % 2) {
        res *= a;
        res %= c;
    }
    a *= a;
    a %= c;
    b = b >> 1;
}
```

Complexity : $O(logB)$

#11444 피보나치 수 6 4

- 행렬의 n제곱? => $O(k^3 \log n)$

```
Matrix pow(lint k) {
    if (!k) return identity();
    if (k % 2) return pow(k - 1) * (*this);
    Matrix half = pow(k / 2);
    return half * half;
}
```

Graphs and Matrices

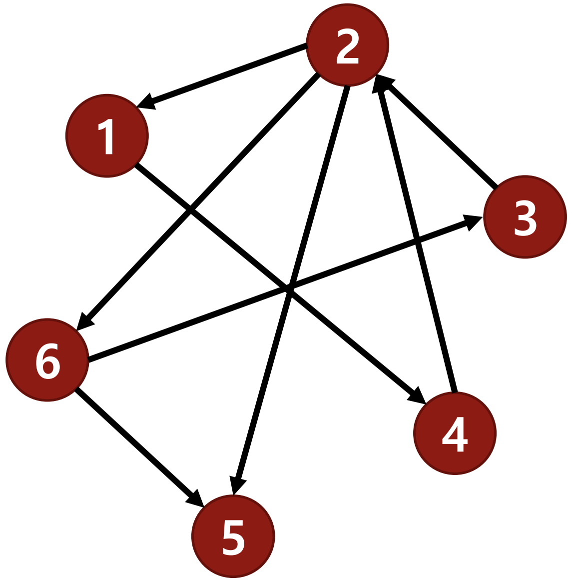
- 가중치 없는 그래프G을 나타낸 인접행렬 M에 대하여

M^n : 노드 쌍 (a, b)에 대하여 n번의 이동을 거쳐 $a \to b$로 가는 경우의 수

Graphs and Matrices

ex)

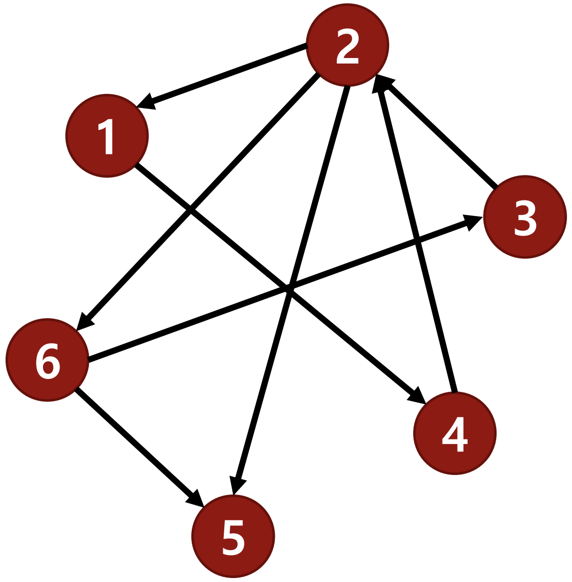
$$M = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$



Graphs and Matrices

ex)

$$M = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$



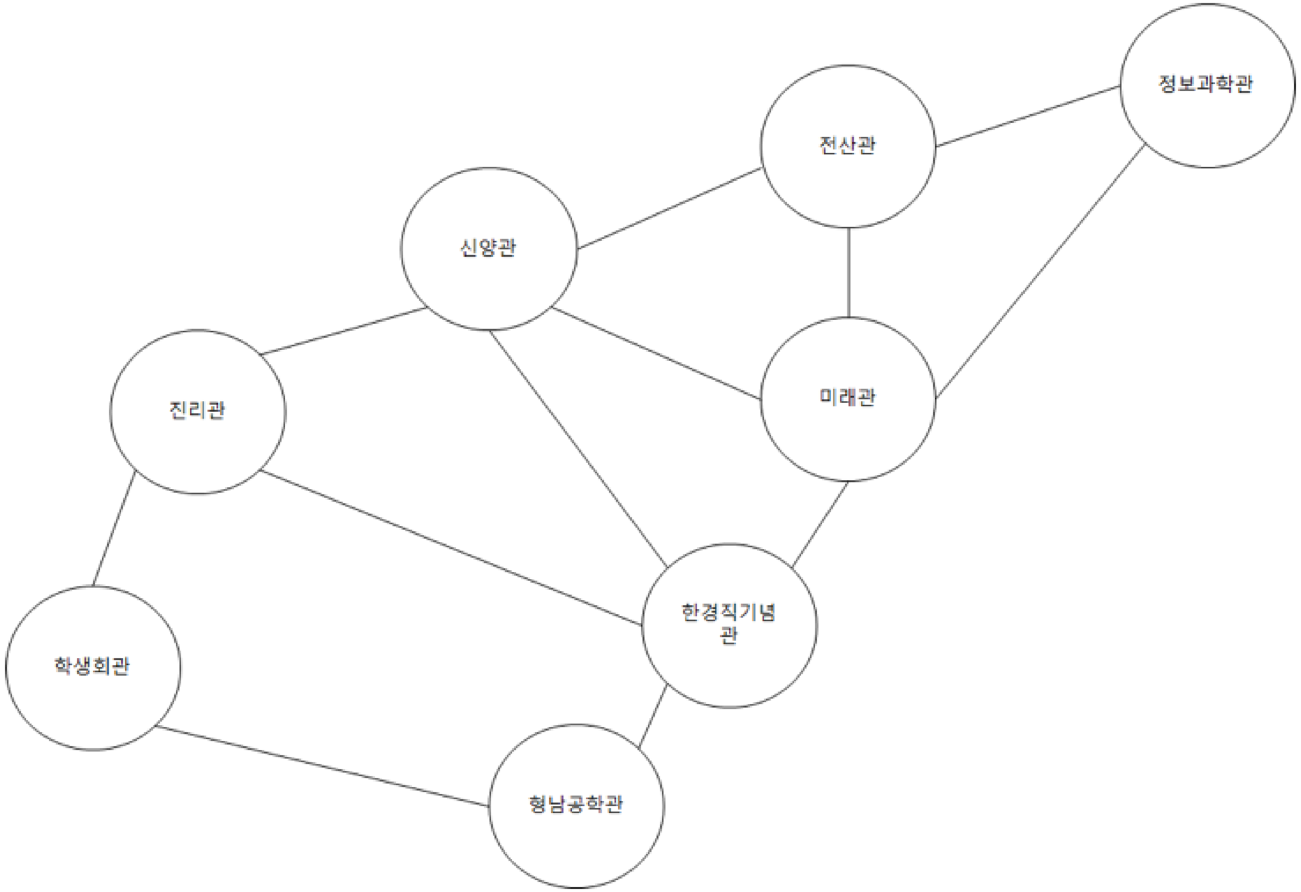
$$M^4 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Graphs and Matrices

- Generizing formula

$$AB[i,j] = \sum_{k=1}^{n} (A[i,k] \cdot B[k,j])$$

#12849 본대 산책 1



#12849 본대 산책 1

- $D \leq 100{,}000$이므로 주어진 그림을 인접행렬에 잘 표현한 후 D제곱.

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

주의 : 틀렸을수도 있습니다.

Problem Set

- 4 11444 피보나치 수 6
- 3 9254 역행렬
- 4 14289 본대산책 3
- 4 2099 The game of death
- ? 16467 병아리의 변신은 무죄
- 3 11049 행렬 곱셈 순서
- 4 17272 리그 오브 레전설 (Large)
- 4 1533 길의 개수
- 3 17401 일하는 세포
- 4 1117D Magic Gems
- 5 15572 블록 4
- 3 3606 Cellular Automaton

Last week's Solution

- 4 11375 열혈강호 karp, B-match
- 2 1799 비숍
- 4 6086 Total Flow
- 4 2188 축사 배정
- 4 11376 열혈강호 2
- 3 1017 소수 쌍
- 3 3295 단방향 링크 네트워크
- 4 14750 Jerry and Tom
- 3 8992 집기 게임