Exponentiation by Squaring

2020 Winter 중급

20141284 이기현

- 이기현
- BOJ - sbrus_1213
- 수학 / 컴퓨터공학
- 2019 SPC Champion 부문 대상
- 2020 UCPC 본선 및 ICPC 본선 입상 (목표)

거듭제곱의 시간복잡도

- $3^5 = 3 * 3 * 3 * 3 * 3 = 243$

- $x^n = ?$

- $3^5 = 3 * 3 * 3 * 3 * 3 = 243$

- $x^n = x * x * x * \ldots * x$

- n번의 곱하기 연산
- 거듭제곱의 시간복잡도 = $O(N)$

```
int pow(int x, int n) {
    int res = 1;
    for (int i = 0; i < n; ++i)
        res *= x;
    return res;
}
```

#1629 곱셈

- A^B 를 C로 나눈 나머지를 구하라
- $1 \leq A, B, C \leq 2{,}147{,}483{,}647$

#1629 곱셈 3

- A^B 를 C로 나눈 나머지를 구하라
- $1 \leq A, B, C \leq 2{,}147{,}483{,}647$

- O(N) 으로 B번 곱하면 -> TLE

#1629 곱셈 3

- 사전지식 1 – 2진수 표현

$$19 = 10011_2$$

$$16 + 2 + 1 = 10000_2 + 10_2 + 1_2$$

- 사전지식 2 – 지수 법칙

$$x^{a+b} = x^a * x^b$$

#1629 곱셈 3

- 사전지식 3 – modular operation의 속성

$$(a + b)\,mod\,p = (a\,mod\,p + b\,mod\,p\,)\,mod\,p$$

$$(a - b)\,mod\,p = (a\,mod\,p - b\,mod\,p\,)\,mod\,p$$

$$(a * b)\,mod\,p = (a\,mod\,p * b\,mod\,p\,)\,mod\,p$$

#1629 곱셈

- 종합해보자.

- $x^{19} \, mod \, p$

#1629 곱셈

- 종합해보자.
- $x^{19}\, mod\, p = x^{10011_2}\, mod\, p$

#1629 곱셈

- 종합해보자.

- $x^{19}\, mod\, p = x^{10011_2}\, mod\, p$

$$= x^{10000_2+10_2+1_2}\, mod\, p$$

#1629 곱셈 3

- 종합해보자.

- $x^{19} \, mod \, p = x^{10011_2} \, mod \, p$

$$= x^{10000_2+10_2+1_2} \, mod \, p$$

$$= x^{16} * x^2 * x^1 \, mod \, p$$

#1629 곱셈 3

- 종합해보자.

- $$\begin{aligned} x^{19}\,mod\,p &= x^{10011_2}\,mod\,p \\ &= x^{10000_2+10_2+1_2}\,mod\,p \\ &= x^{16} * x^{2} * x^{1}\,mod\,p \\ &= \left((x^{16}\,mod\,p) * (x^{2}\,mod\,p) * (x^{1}\,mod\,p)\right) mod\,p \end{aligned}$$

#1629 곱셈

- 지수의 이진수 표현
  - $\mathrm{b} = 19 \rightarrow 10011_2$
- 지수의 LSB 부터 계산
  - $res = a^1 * a^2 * a^{16}$
- Modular operation
  - 5, 6 line , 매 단계에서 modulo 처리

```
1  int pow(int a, int b, int c) {
2      int res = 1;
3      while (b) {
4          if (b % 2)
5              res = (res * a) % c;
6          a = (a * a) % c;
7          b >>= 1;
8      }
9      return res;
10 }
```

#1629 곱셈

- 지수의 이진수 표현
  - $\mathrm{b} = 19 \rightarrow 10011_2$
- 지수의 LSB 부터 계산
  - $res = a^1 * a^2 * a^{16}$
- Modular operation
  - 5, 6 line , 매 단계에서 modulo 처리
- 시간 복잡도 : $O(\log N)$

```
1  int pow(int a, int b, int c) {
2      int res = 1;
3      while (b) {
4          if (b % 2)
5              res = (res * a) % c;
6          a = (a * a) % c;
7          b >>= 1;
8      }
9      return res;
10 }
```

Fermat’s little theorem

- p 가 소수, a 가 정수라고 하자. 페르마 소정리에 의해 다음이 성립한다.

$$a^p \equiv a \pmod{p}$$

Fermat’s little theorem

- p 가 소수, a 가 정수라고 하자. 페르마 소정리에 의해 다음이 성립한다.

$$a^p \equiv a \pmod p$$

- p ≠ a 일 때, 다음도 성립 가능하다.

$$a^{p-1} \equiv 1 \pmod p$$

- 모듈로 역원(Modular inverse)

$$a^{-1} \equiv a^{p-2} \pmod p$$

#11401 이항 계수 3

- $\binom{n}{k}$ 를 1,000,000,007 로 나눈 나머지를 구하여라.
- $1 \le n \le 4{,}000{,}000$, $0 \le k \le n$

#11401 이항 계수 3

- $\binom{n}{k}$ 를 1,000,000,007 로 나눈 나머지를 구하여라.
- $1 \leq n \leq 4{,}000{,}000$, $0 \leq k \leq n$

- Dynamic programming

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

$O(n^2)$ 의 시간 복잡도 -> TLE

#11401 이항 계수 3

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \equiv ? \pmod{p}$$

#11401 이항 계수 3

- modular operation의 속성

$$(a + b)\ mod\ p = (a\ mod\ p + b\ mod\ p\ )\ mod\ p$$

$$(a - b)\ mod\ p = (a\ mod\ p - b\ mod\ p\ )\ mod\ p$$

$$(a * b)\ mod\ p = (a\ mod\ p * b\ mod\ p\ )\ mod\ p$$

$$(a/b)\ \ mod\ p = ?$$

#11401 이항 계수 3

- modular operation의 속성

$$(a+b)\, mod\, p = (a\, mod\, p + b\, mod\, p\,)\, mod\, p$$

$$(a-b)\, mod\, p = (a\, mod\, p - b\, mod\, p\,)\, mod\, p$$

$$(a*b)\, mod\, p = (a\, mod\, p * b\, mod\, p\,)\, mod\, p$$

$$(a/b) mod\, p = (a*b^{-1})\, mod\, p = (a\ *b^{p-2})\, mod\, p$$

#11401 이항 계수 3

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \equiv ? \pmod{p}$$

$$\frac{n!}{k!\,(n-k)!} \equiv n! * (k!\,(n-k)!)^{p-2} \pmod{p}$$

#11401 이항 계수 3

- 1~n의 팩토리얼 값 memoization
- 시간 복잡도 : $O(N)$

```cpp
const int mod = 1'000'000'007;
long long factorial[4'000'001];

void init(int n) {
    factorial[0] = 1;
    for (int i = 1; i <= n; ++i)
        factorial[i] = (factorial[i - 1] * i) % mod;
}
```

#11401 이항 계수 3

- $\binom{n}{k} \equiv n! * (k!\,(n-k)!)^{p-2} \pmod p$
- 시간 복잡도 : $O(\log p)$

```cpp
long long pow(int x, int n) {
    long long res = 1;
    while (n) {
        if (n % 2)
            res = (res * x) % mod;
        x = (x * x) % mod;
        n >>= 1;
    }
    return res;
}

int binomial(int n, int k) {
    long long res;
    res = (factorial[k] * factorial[n - k]) % mod;
    res = pow(res, mod - 2);
    res = (res * factorial[n]) % mod;
    return res;
}
```

#11401 이항 계수 3

- $\binom{n}{k} \equiv n! * (k!\,(n-k)!)^{p-2} \pmod{p}$
- 시간 복잡도 : $O(\log p)$
- 최종 시간 복잡도 : $O(N)$

```
long long pow(int x, int n) {
    long long res = 1;
    while (n) {
        if (n % 2)
            res = (res * x) % mod;
        x = (x * x) % mod;
        n >>= 1;
    }
    return res;
}

int binomial(int n, int k) {
    long long res;
    res = (factorial[k] * factorial[n - k]) % mod;
    res = pow(res, mod - 2);
    res = (res * factorial[n]) % mod;
    return res;
}
```

#11444 피보나치 수 6

- n번째 피보나치 수를 1,000,000,007로 나눈 나머지를 구하여라.
- $1 \leq n \leq 1{,}000{,}000{,}000{,}000{,}000{,}000$

#11444 피보나치 수 6 4

- n번째 피보나치 수를 1,000,000,007로 나눈 나머지를 구하여라.
- $1 \leq n \leq 1{,}000{,}000{,}000{,}000{,}000{,}000$

- Dynamic programming

$$F_n = F_{n-1} + F_{n-2}$$

$O(n)$ 의 시간 복잡도 -> TLE

#11444 피보나치 수 6

- 피보나치 수의 선형 점화식

$$F_{n+1} = 1 * F_n + 1 * F_{n-1}$$

- 선형 점화식의 행렬 표현

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} * \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

#11444 피보나치 수 6 4

- 선형 점화식의 행렬 표현

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} * \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n * \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

#11444 피보나치 수 6

- matrix 곱셈 연산
- 시간 복잡도 : $O(n^3)$
  (n = matrix의 크기)

```
matrix operator * (const matrix rhs) {
    matrix temp(size);
    for (int i = 0; i < size; ++i)
        for (int j = 0; j < size; ++j)
            for (int k = 0; k < size; ++k) {
                temp.item[i][j] += item[i][k] * rhs.item[k][j];
                temp.item[i][j] %= mod;
            }
    return temp;
}
```

#11444 피보나치 수 6 4

- matrix 거듭제곱
- 시간 복잡도 : $O(n^3 \log t)$
  (t = 지수)

```
matrix exp(ll times) {
    matrix res = matrix().identity(size), tmp(size);

    tmp.item = item;

    while (times) {
        if (times % 2)
            res = res * tmp;
        times /= 2;
        tmp = tmp * tmp;
    }
    return res;
}
```

#11444 피보나치 수 6 4

- $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ 초기화 및 n 제곱
- $\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n * \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

- 시간 복잡도 : $O(\mathrm{n}^3 \log t)$

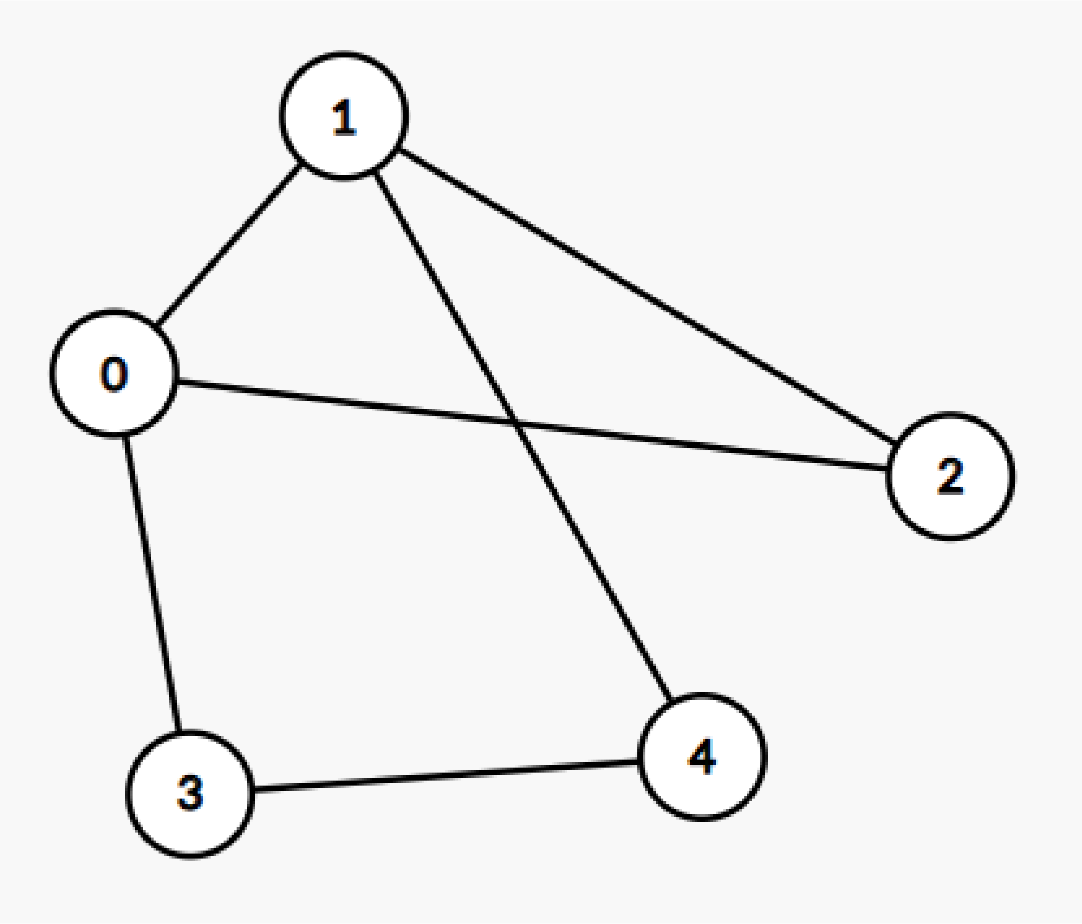
```
int main() {
    ll n;
    cin >> n;

    matrix mat(2), res;
    mat.item[0][0] = mat.item[0][1] = mat.item[1][0] = 1;
    res = mat.exp(n);

    cout << res.item[1][0];
}
```

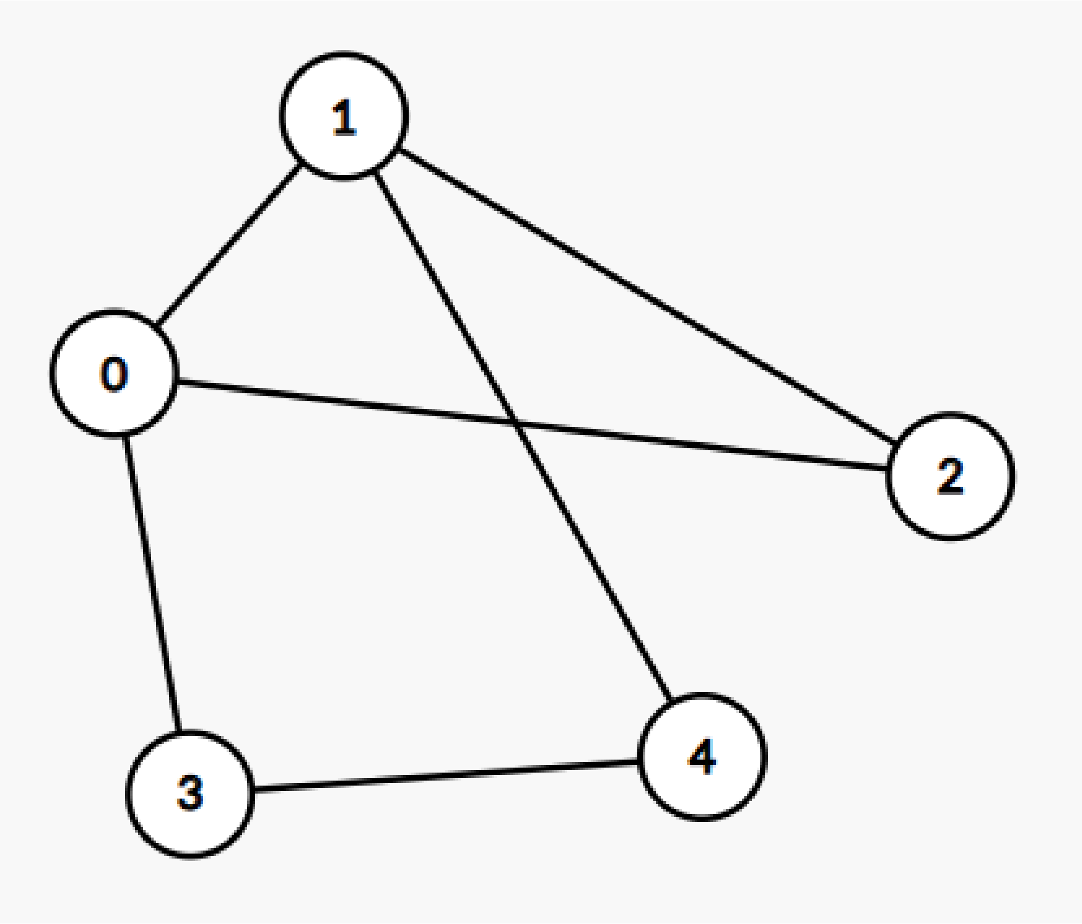
그래프의 인접행렬

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



그래프의 인접행렬

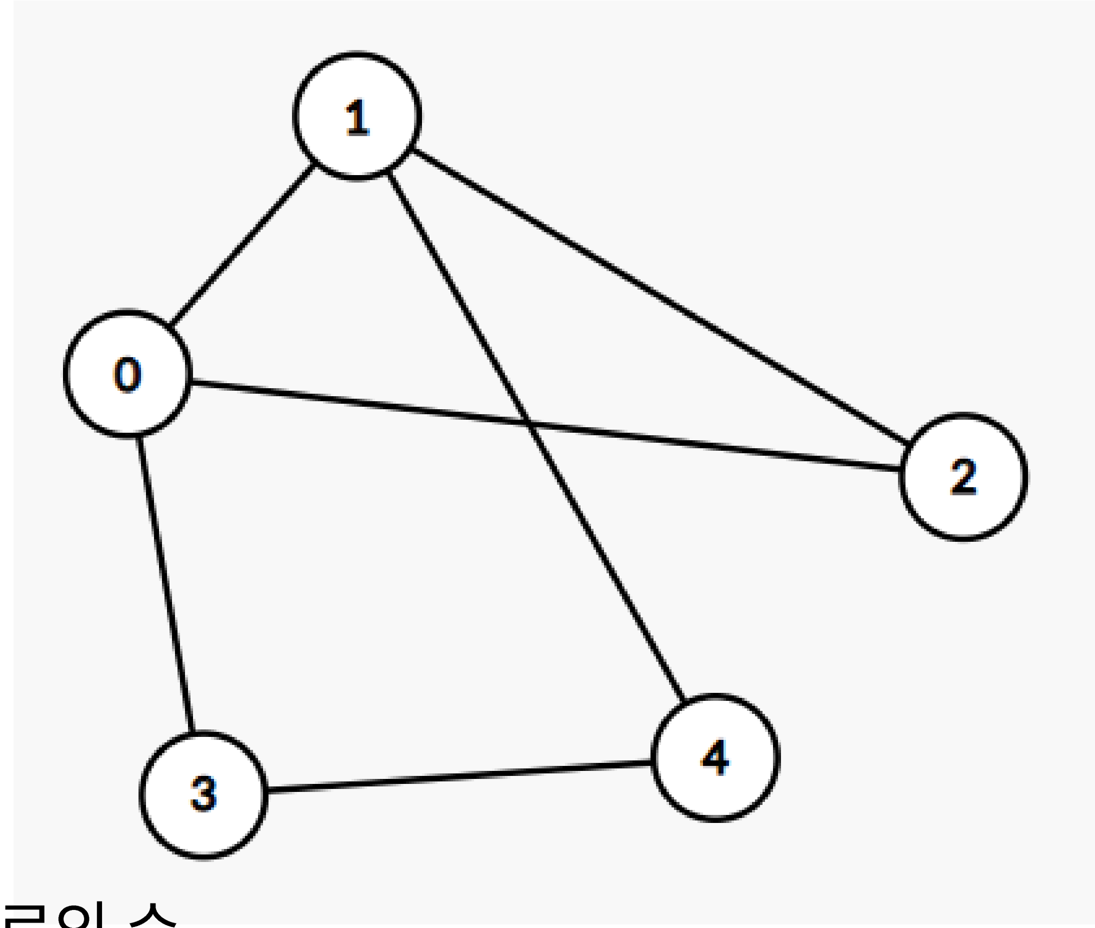
$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$



- M[i][j] = i에서 j로 가는 경로의 수

그래프의 인접행렬

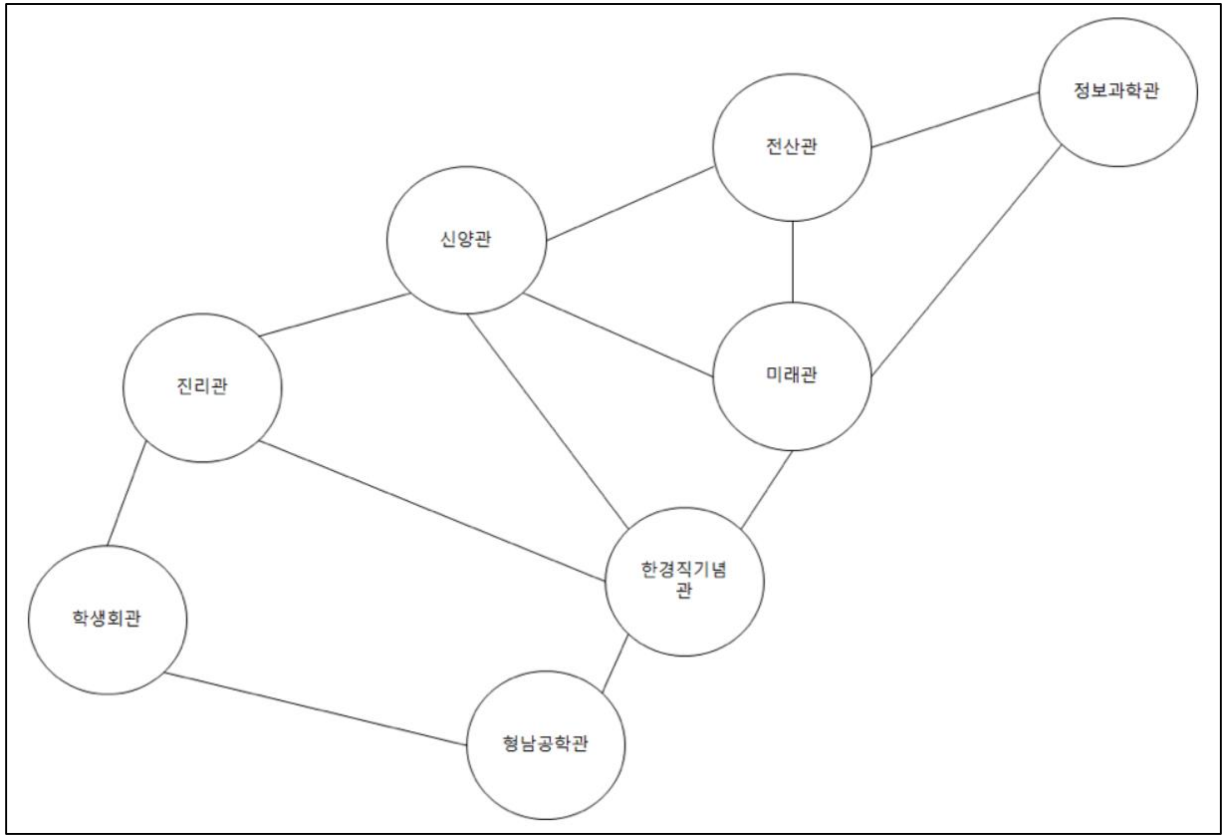
$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$



- M[i][j] = i에서 j로 가는 경로의 수
- M^d[i][j] = d번 이동해서 i에서 j로 가는 경로의 수

#12850 본대 산책 2

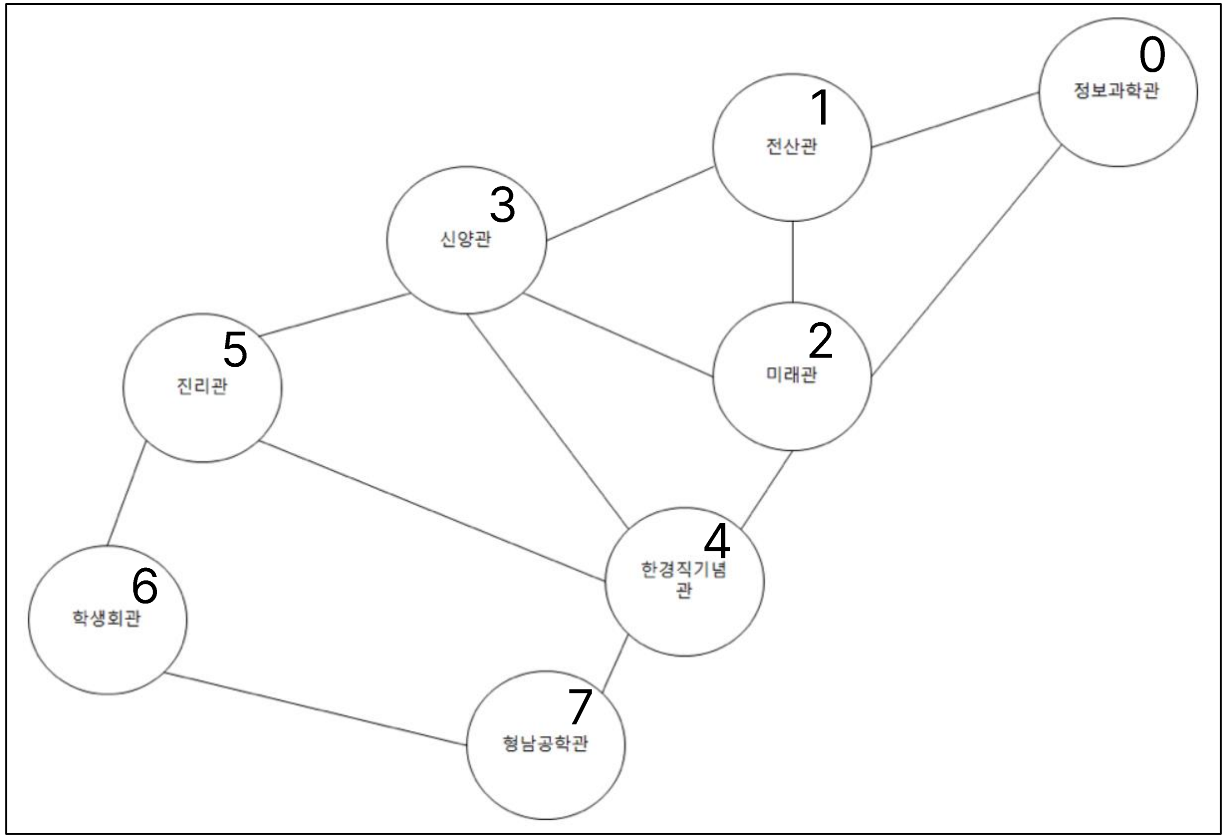
- 8개 건물의 인접 현황
- d번 이동해 정보과학관으로 돌아오는 경로의 개수를 구하여라
- $1 \leq d \leq 1{,}000{,}000{,}000$



#12850 본대 산책 2

1. 인접 행렬 표현

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

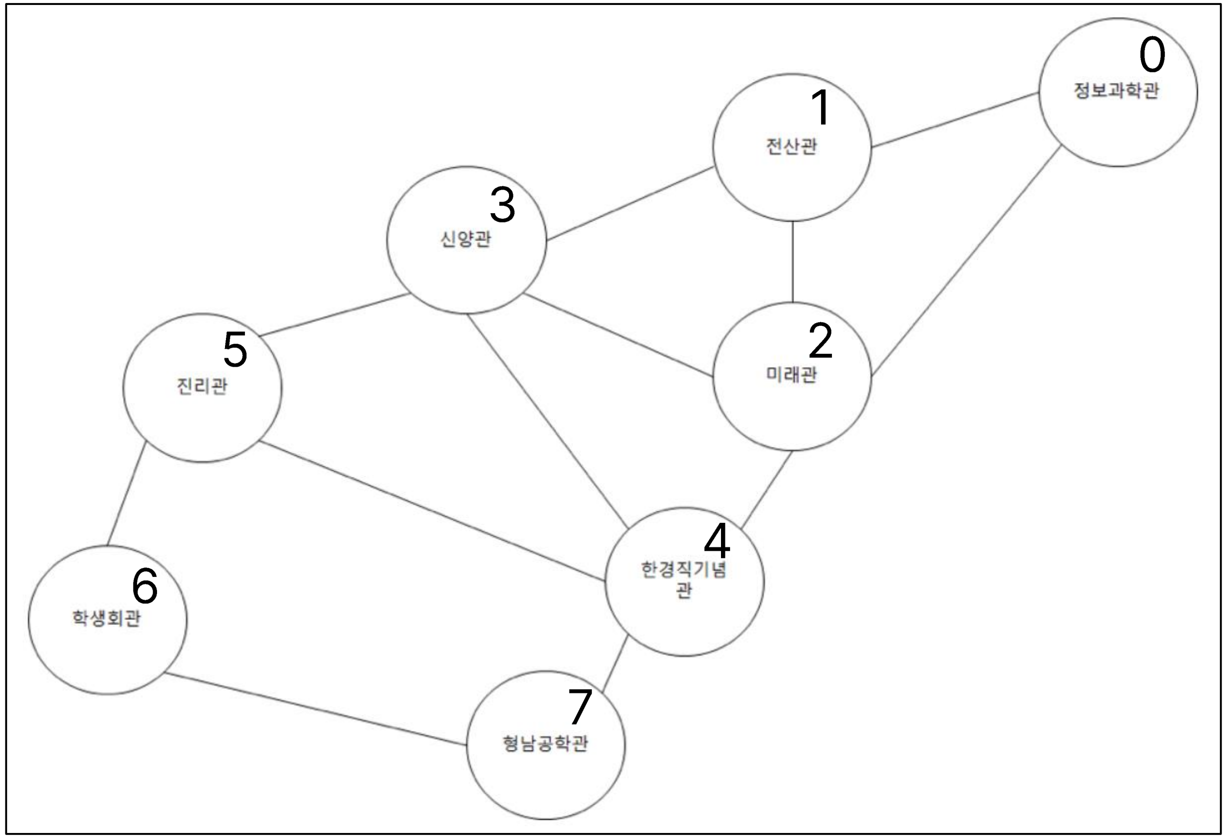


#12850 본대 산책 2

1. 인접 행렬 표현

2. 행렬 거듭제곱

   $M^d[0][0]$ = d번 이동해 정보과학관에 도착하는 경로의 수

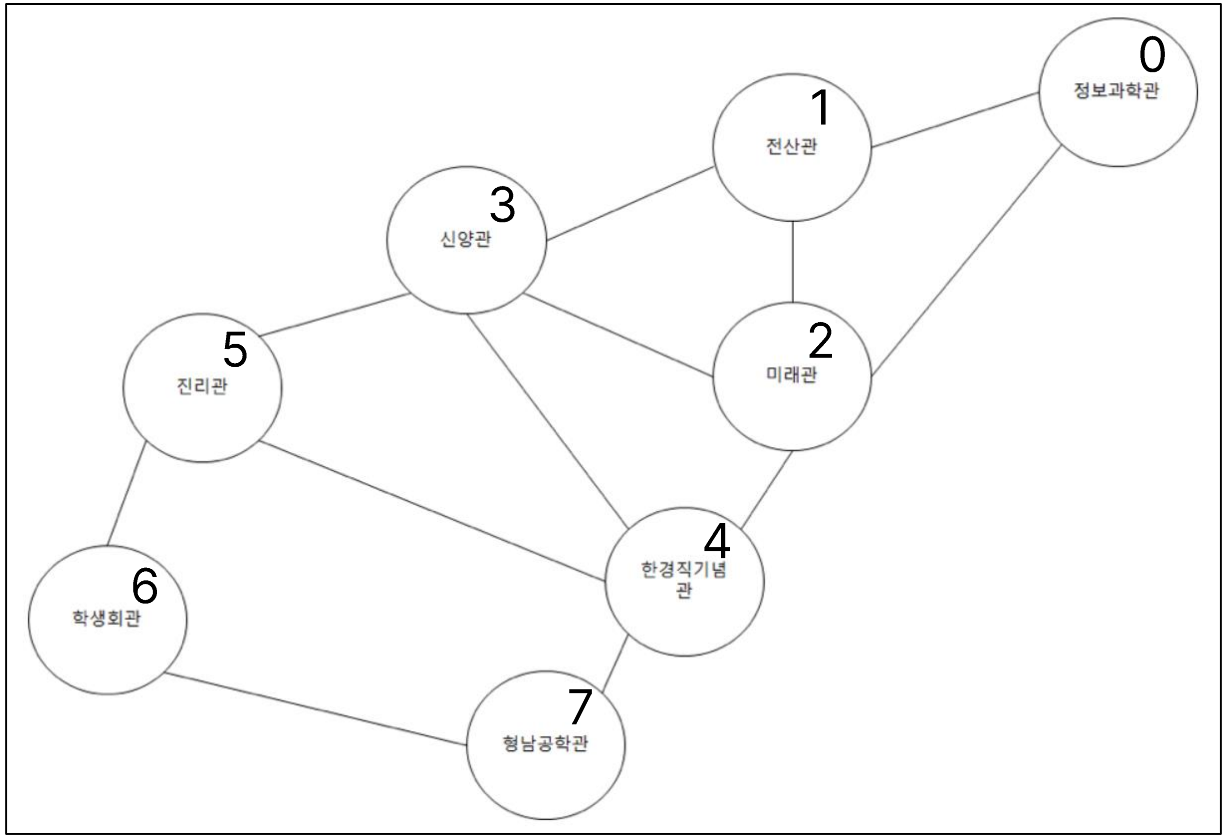


#12850 본대 산책 2 1

1. 인접 행렬 표현

2. 행렬 거듭제곱
   $M^d[0][0]$ = d번 이동해 정보과학관에 도착하는 경로의 수

3. 시간 복잡도 : O(8^3 log d)
   (코드는 생략!)



#17272 리그 오브 레전설 (Large)

- N초 동안 싸움 / A, B 스킬 사용 가능
- A 스킬 – 1초 소모 / B 스킬 – M초 소모
- A 스킬과 B 스킬의 사용 조합의 개수를 1,000,000,007로 나눈 나머지를 구하여라.
  (AB, BA 와 같이 스킬 사용 순서가 다르면 따로 세어준다.)
- $1 \leq N \leq 10^{18}, 2 \leq M \leq 100$

#17272 리그 오브 레전설 (Large)

1. dp 식 찾기

#17272 리그 오브 레전설 (Large)

1. dp 식 찾기

$$dp[1] \sim dp[m-1] = 1, dp[m] = 2$$

$$dp[n] = dp[n-1] + dp[n-m]$$

#17272 리그 오브 레전설 (Large)

1. dp 식 찾기

$$dp[1] \sim dp[m-1] = 1, dp[m] = 2$$

$$dp[n] = dp[n-1] + dp[n-m]$$

2. 선형 점화식의 행렬 표현 (m x m 사이즈의 행렬)

#17272 리그 오브 레전설 (Large)

1. dp 식 찾기

$$dp[1] \sim dp[m-1] = 1, dp[m] = 2$$

$$dp[n] = dp[n-1] + dp[n-m]$$

2. 선형 점화식의 행렬 표현 (m x m 사이즈의 행렬)

$$\begin{pmatrix} dp[n] \\ dp[n-1] \\ dp[n-2] \\ \dots \\ dp[n-m+1] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ & & & \dots & & \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix} * \begin{pmatrix} dp[n-1] \\ dp[n-2] \\ dp[n-3] \\ \dots \\ dp[n-m] \end{pmatrix}$$

#17272 리그 오브 레전설 (Large)

3. 행렬의 거듭제곱

#17272 리그 오브 레전설 (Large)

3. 행렬의 거듭제곱

$$\begin{pmatrix} dp[n] \\ dp[n-1] \\ dp[n-2] \\ \dots \\ dp[n-m+1] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ & & & \dots & & \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}^{n-m} * \begin{pmatrix} dp[m] \\ dp[m-1] \\ dp[m-2] \\ \dots \\ dp[1] \end{pmatrix}$$

#17272 리그 오브 레전설 (Large)

3. 행렬의 거듭제곱

$$\begin{pmatrix} dp[n] \\ dp[n-1] \\ dp[n-2] \\ \dots \\ dp[n-m+1] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ & & & \dots & & \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}^{n-m} * \begin{pmatrix} dp[m] \\ dp[m-1] \\ dp[m-2] \\ \dots \\ dp[1] \end{pmatrix}$$

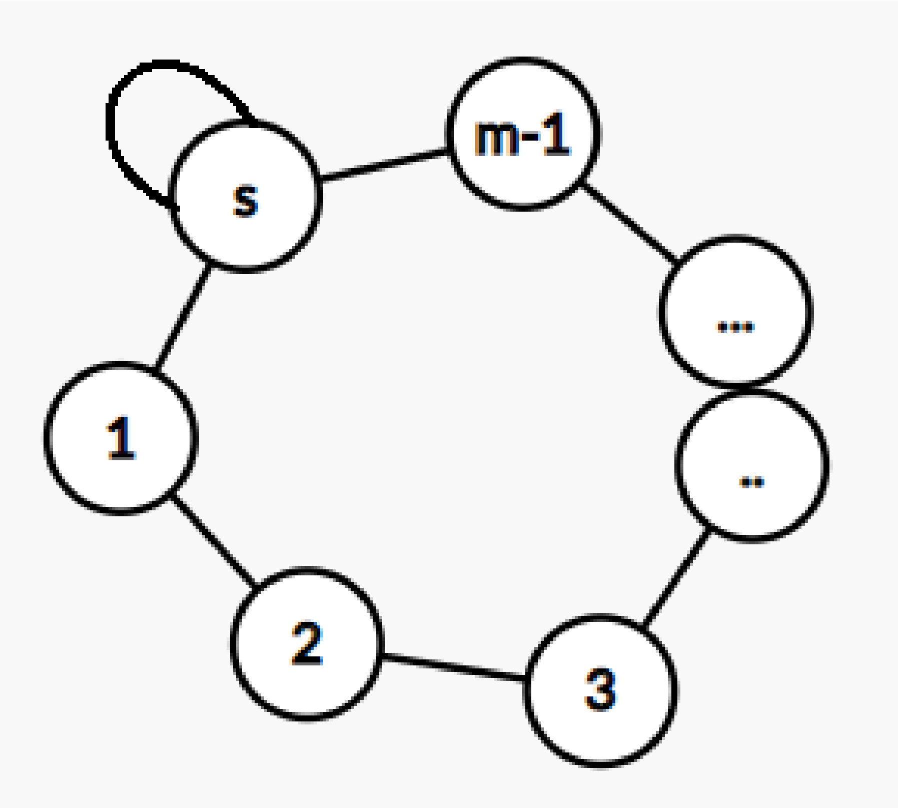
4. 시간 복잡도 : $O(M^3 \log N)$

(코드는 생략,,)

#17272 리그 오브 레전설 (Large)

- [다른 풀이] – 그래프 이용

- 1개의 시작노드, m-1개의 노드
- 각 edge의 가중치 = 1
- n번 이동 후, 다시 시작점으로 돌아오는 경로의 수



문제 추천

- 3 1629 곱셈
- 4 10830 행렬 제곱
- 4 11444 피보나치 수 6
- 3 2099 The game of death
- 1 11401 이항 계수 3
- 1 12850 본대산책 2
- 1 14289 본대산책 3
- 1 13976 타일 채우기 2
- 1 17272 리그 오브 레전설 (Large)
- 5 18117 분수
- 5 13328 Message Passing
- 5 17401 일하는 세포
- 3 17415 Huge Integer!
- 2 3606 Cellular Automaton