선형대수 팀플 프로젝트 과제 2 – 행렬의 멱승

G조 정유진, 문다인, 아크바이 아미라, 유환희, 홍진선

행렬의 멱승

- 정방행렬 A에 대해서, 자기 자신과 반복해서 곱함으로써 음이 아닌 정수 차수의 멱승을 정의할 수 있다.
- 만약 m이 양의 정수이고 A가 K x K 행렬이라면,

$$A^m = \underbrace{\{A \cdot A \cdots A\}}_{m\text{번}} \text{로 정의된다.}$$

- 또한 우리는 다음과 같은 약속을 채택한다.

$$A^0 = I$$

- I는 K x K 단위행렬이다.

빨리 계산하는 방법

- Binary Exponentiation 은 분할정복(divide and conquer) 기법을 이용하여 거듭제곱을 계산하는 알고리즘이며 조건은 다음과 같다.

$$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$$

- 이 알고리즘은 다음과 같은 거듭제곱의 성질을 이용한다.

$$a^{b+c} = a^b \cdot a^c \text{ and } a^{2b} = a^b \cdot a^b = (a^b)^2.$$

- 지수 m을 이진수로 표현하면, A^m은 몇 개의 A^{2^k}항의 곱으로 분해될 수 있다. 따라서 A^m을 직접 m번 곱하는 대신, 행렬의 제곱을 반복하여 계산 횟수를 크게 줄일 수 있다.
- Binary Exponentiation 알고리즘은 다음과 같은 절차로 동작한다.

1. 결과 행렬을 단위행렬 I로 초기화한다.
2. 지수 m이 0이 될 때까지 반복한다.
3. m이 홀수이면, 결과 행렬에 현재 행렬을 곱한다.
4. 현재 행렬을 자기 자신과 곱하여 제곱한다.
5. m을 2로 나눈 몫으로 갱신한다.
6. 이 과정에서 지수 m은 매 단계마다 절반으로 줄어들기 때문에, 필요한 행렬 곱셈의 횟수는 $O(\log m)$에 불과하다.

- 다음과 같은 재귀적 형태로도 표현할 수 있다.

$$a^n = \begin{cases} 1 & \text{if } n == 0 \\ \left(a^{\frac{n}{2}}\right)^2 & \text{if } n > 0 \text{ and } n \text{ even} \\ \left(a^{\frac{n-1}{2}}\right)^2 \cdot a & \text{if } n > 0 \text{ and } n \text{ odd} \end{cases}$$

- 시간 복잡도 역시 기존 반복 행렬곱 방식은 $O(m)$인것에 비해 Binary Exponentiation 방식은 $O(logm)$이라 훨씬 효율적이다.
- Binary Exponentiation은 곱셈 연산이 결합법칙을 만족하면 성립하는데 행렬 곱셈은 결합법칙을 만족하므로 행렬 곱셈에서도 성립한다.

테스트 케이스 결과

1. 기본 단위 행렬

```
행렬 크기 n (정방행렬 n x n): 2
2x2 행렬 A의 원소를 행 단위로 입력하세요. (공백으로 구분)
1번째 행: 1 0
2번째 행: 0 1
몇 승(m): 5

================ 결과 ================
m = 5

[1) 기존 방식(반복 행렬곱)]
1 0
0 1
걸린 시간: 0.000037 sec

[2) Binary Exponentiation(분할정복)]
1 0
0 1
걸린 시간: 0.000018 sec
====================================
```

2. 0승인 경우

```
행렬 크기 n (정방행렬 n x n): 3
3x3 행렬 A의 원소를 행 단위로 입력하세요. (공백으로 구분)
1번째 행: 2 1 0
2번째 행: 0 1 3
3번째 행: 4 0 1
몇 승(m): 0

================ 결과 ================
m = 0

[1) 기존 방식(반복 행렬곱)]
1 0 0
0 1 0
0 0 1
걸린 시간: 0.000017 sec

[2) Binary Exponentiation(분할정복)]
1 0 0
0 1 0
0 0 1
걸린 시간: 0.000014 sec
====================================
```

3. 일반 행렬 (2승)

```
행렬 크기 n (정방행렬 n x n): 2
2x2 행렬 A의 원소를 행 단위로 입력하세요. (공백으로 구분)
1번째 행: 1 2
2번째 행: 3 4
몇 승(m): 2

================ 결과 ================
m = 2

[1) 기존 방식(반복 행렬곱)]
7 10
15 22
걸린 시간: 0.000041 sec

[2) Binary Exponentiation(분할정복)]
7 10
15 22
걸린 시간: 0.000026 sec
====================================
```

4. 일반 행렬 (20승)

```
행렬 크기 n (정방행렬 n x n): 2
2x2 행렬 A의 원소를 행 단위로 입력하세요. (공백으로 구분)
1번째 행: 1 1
2번째 행: 1 0
몇 승(m): 20

================ 결과 ================
m = 20

[1) 기존 방식(반복 행렬곱)]
10946 6765
6765 4181
걸린 시간: 0.000097 sec

[2) Binary Exponentiation(분할정복)]
10946 6765
6765 4181
걸린 시간: 0.000051 sec
====================================
```

5. 일반 행렬 (50승)

```
행렬 크기 n (정방행렬 n x n): 2
2x2 행렬 A의 원소를 행 단위로 입력하세요. (공백으로 구분)
1번째 행: 1 2
2번째 행: 3 4
몇 승(m): 50

================ 결과 ================
m = 50

[1) 기존 방식(반복 행렬곱)]
7693492221386182773220971348272087O4 11212704116760747491012254977260257 28
168190561751411197607788565691262566 4 2451254839652729958252077612387008512
걸린 시간: 0.000077 sec

[2) Binary Exponentiation(분할정복)]
769349222138617982174191955474382848 11212704116760746015272729080496128 00
168190561751411197607788565691262566 4 2451254839652729958252077612387008512
걸린 시간: 0.000026 sec
====================================
```

- Binary Exponentiation 방법이 반복 행렬곱 방식보다 빠름을 관찰할 수 있다.

참고 문헌

- 행렬의 멱승
  - https://www.statlect.com/matrix-algebra/matrix-power
- 빨리 계산하는 방법
  - https://cp-algorithms.com/algebra/binary-exp.html
  - https://www.freecodecamp.org/news/binary-exponentiation-algorithm-explained-with-examples/