Chained Matrix Multiplication Algorithm

소프트웨어학부

20170294 박해영

< 목 차 >

1. 목표

동적계획으로 해답을 구할 수 있는 문제인 연쇄 행렬곱셈의 알고리즘을 이해하고자 한다. 연쇄 행렬곱셈의 곱셈 횟수의 최수치를 구하기 위해서 부분사례에 대한 최적해를 포함해야 하므로, 해당 알고리즘에 최적의 원칙이 적용됨을 알고리즘 분석과정을 통해 알고자 하며, 이에 따른 시간 복잡도를 도출해보고자 한다.

2. Problem & Input / Output

* **Problem** : n개의 행렬을 곱하는데 필요한 기본적인 곱셈의 횟수의 최소치를 결정하고,
그 최소치를 구하는 순서를 결정하라
* **Input** : 행렬의 수 n, 행렬의 규모를 나타내는 d[0-n]
* **Output** : 기본적으로 곱셈의 횟수의 최소치를 나타내는 minmult;
최적의 순서를 얻을 수 있는 배열 P & 최적의 순서

3. 구현 Language & 사용 Tool

* **구현 언어** : C language
* 사용 Tool : Visual Studio 2017

4. 교재의 입력 데이터 테스트

1) Ex3.5

```
Input tho number of matrices :  6
1번째 행렬의 size (m X n) : 5 2
2번째 행렬의 size (m X n) : 2 3
3번째 행렬의 size (m X n) : 3 4
4번째 행렬의 size (m X n) : 4 6
5번째 행렬의 size (m X n) : 6 7
6번째 행렬의 size (m X n) : 7 8



diagonal = 6 일 때, M 행렬의 결과
  0   0   0   0   0   0   0
  0   0  30  64 132 226 348
  0   0   0  24  72 156 268
  0   0   0   0  72 198 366
  0   0   0   0   0 168 392
  0   0   0   0   0   0 336
  0   0   0   0   0   0   0

diagonal = 6 일 때, P 행렬의 결과
  0   0   0   0   0   0   0
  0   0   1   1   1   1   1
  0   0   0   2   3   4   5
  0   0   0   0   3   4   5
  0   0   0   0   0   4   5
  0   0   0   0   0   0   5
  0   0   0   0   0   0   0

Minimum Multiplication of A1-A6 : 348
Order of Minimun Multiplocation : (A1((((A2A3)A4)A5)A6))
```

5. 자작 입력 데이터 생성 & 알고리즘의 과정 손계산

자작 데이터 생성

행렬 개수 n = 5,

배열생성 ; d[6], M[6][6], P[6][6]

d행렬 입력

A_1	A_2	A_3	A_4	A_5
3×2	2×4	4×6	6×2	2×3

모두 0으로 초기화

※ minmult(5, d[ ], P[ ][ ])

diagonal = 1 ~ 4.

i = 1 ~ 5 − diagonal

k = i ~ j−1

반복문 실행

① diagonal = 1

i = 1 , j = 1+1 = 2

k = 1

M[1][1]+M[2][2]+d[0]×d[1]×d[2] < M[1][2]

|| M[1][2] == 0

⇒ 0+0+3×2×4 < 0 || M[1][2]=0 ⇒True

M[1][2] = 0+0+3×2×4 = 24

P[1][2] = 1

대각선이 1인경우, 그냥행렬 2개의 곱 ($A_1 \times A_2$)

이므로 M[i][j]의 값이 곱셈값을 적용된다

(즉, 행렬 2개의 곱은 그 행렬곱셈수가 최적이다)

M =

	0	1	2	3	4	5
0	0	0	0	0	0	0
1		0	24	0	0	0
2			0	48	0	0
3				0	48	0
4					0	36
5						0

P =

	0	1	2	3	4	5
0	0	0	0	0	0	0
1		0	1	0	0	0
2			0	2	0	0
3				0	3	0
4					0	4
5						0

② diagonal = 2.

i = 1 , j = 1+2 = 3

k = 1

M[1][1]+M[2][3]+d[0]d[1]d[3] < M[1][3]

0 + 48 + 3×2×6 < 0 || M[1][3]=0.

(True)

M[1][3] = 0+48+3×2×6 = 84

P[1][3] = 1

k=2

M[1][2]+M[2][3]+d[0]d[2]d[3] < M[1][3]

24 + 48 + 3×4×6 < 84 (false)

i=2 , j= 2+2 = 4

k=2

M[2][2]+M[3][4]+d[1]d[2]d[4] < M[2][4]

0 + 48 + 2×4×2 < 0 || M[2][4] == 0

(True)

M[2][4] = 0+48+2×4×2 = 64

P[2][4] = 2.

k=3

M[2][3]+M[4][4]+d[1]d[3]d[4] < M[2][4]

48 + 0 + 2×6×2 < 64 (False)

※ k루프문이 처음들때는 애초 M[i][j] 값이 0으로 초기화 되어있기 때문에 위친 대입을 하고 그 다음 k 이 대하여 대소 비교를 진행한다

i=3 , j= 3+2 = 5

k=3

M[3][3]+M[4][5]+d[2]·d[3]·d[5] < M[3][5]

0 + 36 + 4×6×3 < 0 || M[3][5] == 0

(True)

M[3][5] = 0+36+4×6×3 = 108

P[3][5] = 3.

k=4

M[3][4]+M[5][5]+d[3]·d[4]·d[5] < M[3][5]

48 + 0 + 4×2×3 < 108 (True).

M[3][5] = 48+0+4×2×3 = 72.

P[3][5] = 4

※ k루프는 diagonal 만큼 도는것을 알수있다

M=
0	0	0	0	0	0
	0	24	84	0	0
		0	48	64	0
			0	48	72
				0	36
					0

P=
0	0	0	0	0	0
	0	1	1	0	0
		0	2	2	0
			0	3	4
				0	4
					0

⇒ 위와 같은 방법으로 진행하였을 때, 계산결과

③ diagonal = 3

i = 1 , j = 1+3 = 4

k = 1

M[1][4] = 76 , P[1][4] = 1

i = 2 , j = 2+3 = 5

k = 2

M[2][5] = 96 , P[2][5] = 2

k = 4

M[2][5] = 76 , P[2][5] = 4

M=
0	0	0	0	0	0
	0	24	84	76	0
		0	48	64	76
			0	48	72
				0	36
					0

P=
0	0	0	0	0	0
	0	1	1	1	0
		0	2	2	4
			0	3	4
				0	4
					0

④ diagonal = 4

i = 1 , j = 1+4 = 5

k = 1

M[1][5] = 94 , P[1][5] = 1

M=
0	0	0	0	0	0
	0	24	84	76	94
		0	48	64	76
			0	48	72
				0	36
					0

P=
0	0	0	0	0	0
	0	1	1	1	1
		0	2	2	4
			0	3	4
				0	4
					0

∴ Minimum Multiplication of A1-A5 : 94

⊛ i 구하는 횟수 = n-diagonal

k 구하는 횟수 = diagonal

$(= (j-1) - i + 1 = \underline{i + diagonal - 1 - i + 1})$

∴ Basic Operation = M array가 계산되는 횟수

$$\sum_{diagonal=1}^{n-1} (n-diagonal) \times diagonal$$

$$= \frac{n(n-1)(n+1)}{2} \in \Theta(n^3)$$

∴ Time Complexity : $\Theta(n^3)$

---

* 최적의 순서 출력

```
Order(1,5)
    K = P[1][5] = 1
    출력 '('
    Order(1,1);
      출력 'A1'
    Order(2,5)
        K = P[2][5] = 4
        출력 '('
        Order(2,4)
            K = P[2][4] = 2
            출력 '('
            Order(2,2)
              출력 'A2'
            Order(3,4)
                K = P[3][4] = 3
                출력 '('
                Order(3,3)
                  출력 'A3'
                Order(4,4)
                  출력 'A4'
                출력 ')'
            출력 ')'
        Order(5,5)
          출력 'A5'
```

출력 ')'

출력 ')'

* 최적의 순서 - 최종결과

⇒ (A1((A2(A3A4))A5))

6. 자작 입력 데이터 테스트

```
Input tho number of matrices :  5
1번째 행렬의 size (m X n) : 3 2
2번째 행렬의 size (m X n) : 2 4
3번째 행렬의 size (m X n) : 4 6
4번째 행렬의 size (m X n) : 6 2
5번째 행렬의 size (m X n) : 2 3



diagonal = 5 일 때, M 행렬의 결과
  0   0   0   0   0   0
  0   0  24  84  76  94
  0   0   0  48  64  76
  0   0   0   0  48  72
  0   0   0   0   0  36
  0   0   0   0   0   0

diagonal = 5 일 때, P 행렬의 결과
  0   0   0   0   0   0
  0   0   1   1   1   1
  0   0   0   2   2   4
  0   0   0   0   3   4
  0   0   0   0   0   4
  0   0   0   0   0   0

Minimum Multiplication of A1-A5 : 94
Order of Minimun Multiplocation : (A1((A2(A3A4))A5))
```