6. 동적 계획 알고리즘 1

한국외국어대학교
고 석 훈

목차

6.1 동적 계획 알고리즘

- 동적 계획(Dynamic Programming) 알고리즘

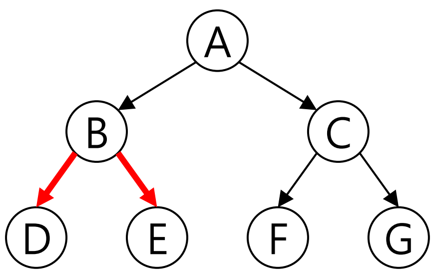
  - 동적 계획 알고리즘은 그리디 알고리즘과 같이 최적화 문제를 해결하는 알고리즘이다.

  - 동적 계획 알고리즘을 적용하기 위해서는 그 문제가 최적 부분 구조의 특성을 갖고 있어야 한다.
    - 최적 부분 구조(optimal substructure): 전체 문제의 최적해가 부분 문제의 최적해로 만들어지는 구조

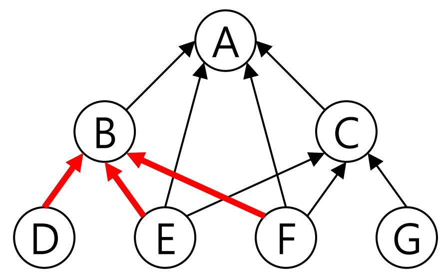
분할 정복 알고리즘 vs 동적 계획 알고리즘

- 분할 정복 알고리즘:

1) A는 B와 C로 분할되고, B는 D와 E로 분할된다.
2) D와 E의 해를 취합하여 B의 해를 구한다.
   단, D, E, F, G는 각각 더 이상 분할할 수 없는 부분 문제들이다.
3) 마찬가지로 F와 G의 해를 취합하여 C의 해를 구하고,
   마지막으로 B와 C의 해를 취합하여 A의 해를 구한다.



분할 정복 알고리즘

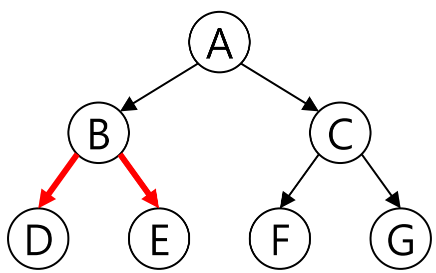


동적 계획 알고리즘

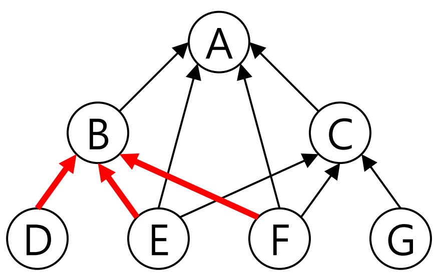
- 동적 계획 알고리즘:

1) 먼저 최소 단위의 부분 문제 D, E, F, G의 해를 각각 구한다.
2) 그 다음 D, E, F의 해를 이용하여 B의 해를 구하고
   E, F, G의 해를 이용하여 C의 해를 구한다.

- 즉, B와 C의 해를 구하는데 E와 F의 해 모두를 이용한다. 반면에 분할 정복 알고리즘은 부분 문제의 해를 중복 사용하지 않는다.

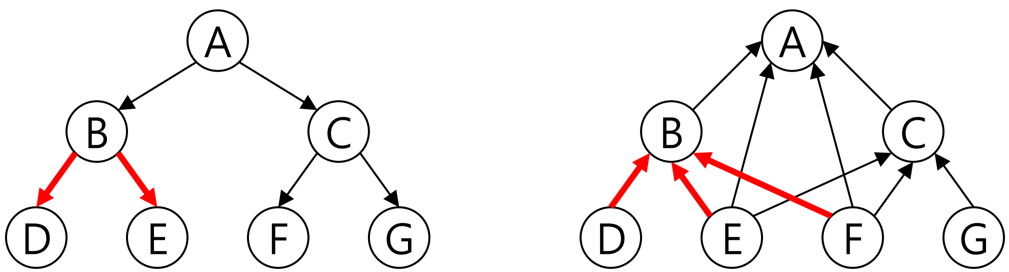


분할 정복 알고리즘



동적 계획 알고리즘

- 동적 계획 알고리즘은 부분 문제들 사이에 의존적 관계가 존재
  - 예를 들면, D, E, F의 해가 B를 해결하는데 사용되는 관계가 있다.
  - 이러한 관계는 문제 또는 입력에 따라 다르고, 대부분의 경우 뚜렷이 보이지 않아서 **함축적 순서(implicit order)**라 한다.

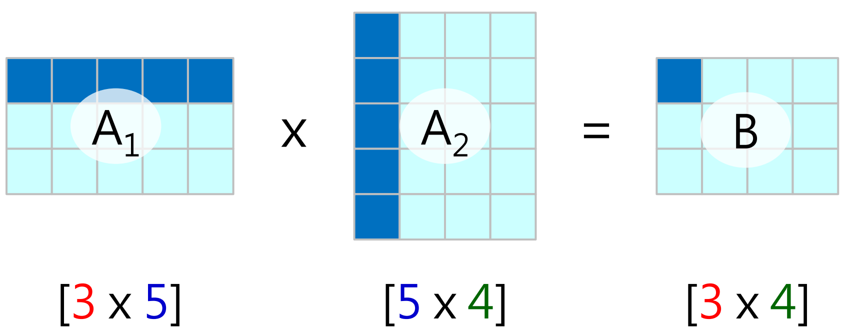


분할 정복 알고리즘 동적 계획 알고리즘

6.2 연속 행렬 곱셈 문제

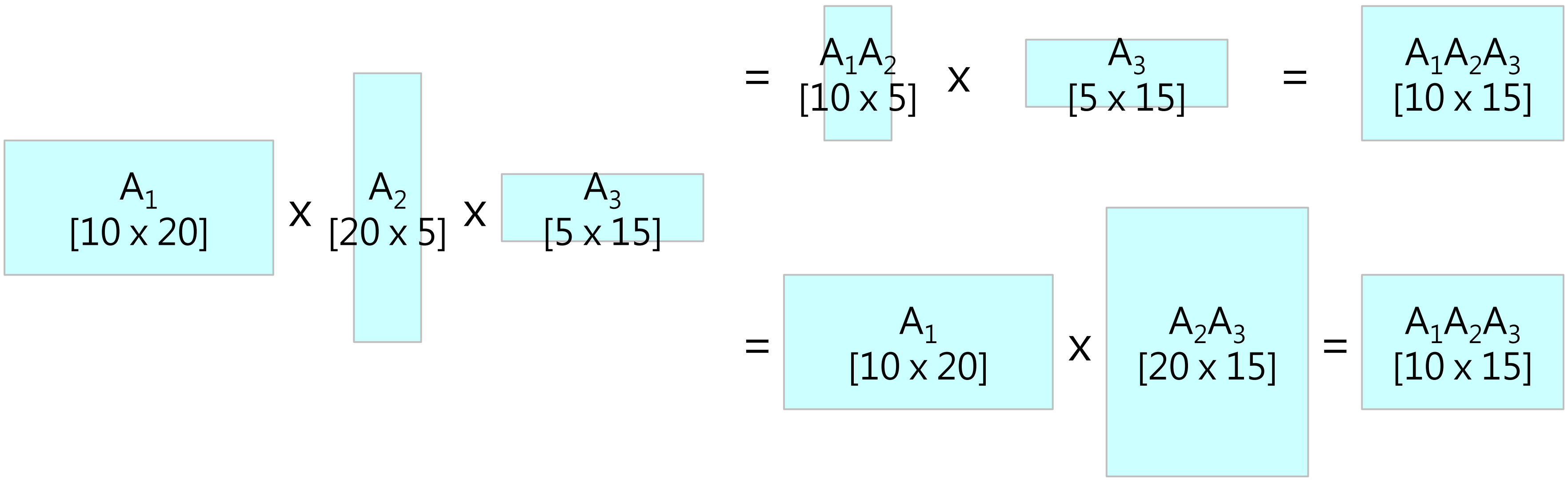
- 연속 행렬 곱셈(Chained Matrix Multiplications) 문제
  - 연속된 행렬들의 곱셈에 필요한 원소간의 **최소 곱셈 횟수**를 찾는 문제

- 행렬의 곱셈
  - [3 x 5] 행렬 A_1과 [5 x 4] 행렬 A_2를 곱한 행렬 B의 크기는 [3 x 4]
  - 이때, 원소간의 곱셈 횟수는 3 x 5 x 4 = 60회이다.

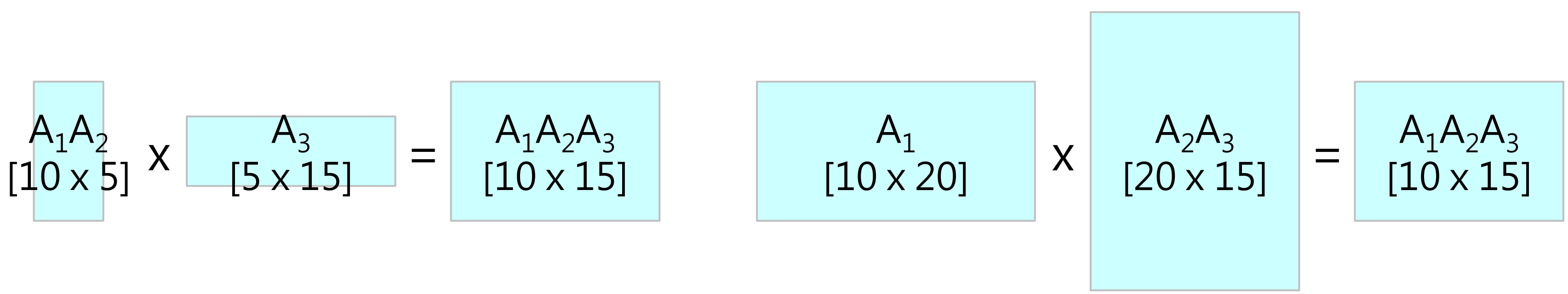


행렬 곱셈

- 3개의 행렬을 곱하는 경우
  - 연속된 행렬의 곱셈에는 결합 법칙이 허용됨
  - $A_1 \times A_2 \times A_3 = (A_1 \times A_2) \times A_3 = A_1 \times (A_2 \times A_3)$



- $A_1[10 \times 20] \times A_2[20 \times 5] \times A_3[5 \times 15]$ 경우
  - $(A_1 \times A_2) \times A_3$ 순서로 곱하는 경우
    - $A_1[10 \times 20] \times A_2[20 \times 5]$ 계산에 원소 간의 곱셈 $10 \times 20 \times 5 = 1{,}000$회 필요
    - $A_1A_2[10 \times 5] \times A_3[5 \times 15]$ 계산에 원소 간의 곱셈 $10 \times 5 \times 15 = 750$회 필요
    - 총 $1{,}000 + 750 = 1{,}750$회의 곱셈 필요
  - $A_1 \times (A_2 \times A_3)$ 순서로 곱하는 경우
    - $A_2[20 \times 5] \times A_3[5 \times 15]$ 계산에 원소 간의 곱셈 $20 \times 5 \times 15 = 1{,}500$회 필요
    - $A_1[10 \times 20] \times A_2A_3[20 \times 15]$ 계산에 원소 간의 곱셈 $10 \times 20 \times 15 = 3{,}000$회 필요
    - 총 $1{,}500 + 3{,}000 = 4{,}500$회의 곱셈 필요



- 연속 행렬 곱셈 문제
  - 동일한 결과를 얻음에도 불구하고, 원소간의 곱셈 횟수가 4,500 - 1,700 = 2,800이나 차이 난다.
  - 따라서, 연속된 행렬을 곱하는데 필요한 원소간의 곱셈 횟수를 최소화시키기 위해서는 적절한 행렬의 곱셈 순서를 찾아야 한다.

부분 문제

- 연속 행렬 곱셈의 부분 문제
  - 주어진 행렬의 순서를 지켜 이웃하는 행렬끼리 반드시 곱해야 한다. 예를 들어, $A_1 \times A_2 \times A_3 \times A_4$를 계산할 때, A_2를 건너뛰고 $A_1 \times A_3$을 먼저 수행할 수 없다.
  - $A_2 \times A_3$은 $A_1 \times A_2 \times A_3$의 부분 문제이자 $A_2 \times A_3 \times A_4$의 부분 문제이며, $A_1 \times A_2 \times A_3 \times A_4$의 부분 문제 이다. 따라서, 큰 문제를 풀기 위해 모든 부분 문제들의 해를 먼저 구해야 한다.

부분 문제 크기	부분 문제	부분 문제 개수
0	A_1 A_2 A_3 A_4	4개
1	$A_1 \times A_2$ $A_2 \times A_3$ $A_3 \times A_4$	3개
2	$A_1 \times A_2 \times A_3$ $A_2 \times A_3 \times A_4$	2개
3	$A_1 \times A_2 \times A_3 \times A_4$	1개

부분 문제 분할

$A_1 \times A_2 \times A_3 \times A_4$

minCount{$A_1 \times A_2A_3A_4$, $A_1A_2 \times A_3A_4$, $A_1A_2A_3 \times A_4$}

$A_1 \times A_2 \times A_3$

minCount{$A_1 \times A_2A_3$, $A_1A_2 \times A_3$}

$A_2 \times A_3 \times A_4$

minCount{$A_2 \times A_3A_4$, $A_2A_3 \times A_4$}

$A_1 \times A_2$

minCount{$A_1 \times A_2$}

$A_2 \times A_3$

minCount{$A_2 \times A_3$}

$A_3 \times A_4$

minCount{$A_3 \times A_4$}

A_1

minCount{A_1}

A_2

minCount{A_2}

A_3

minCount{A_3}

A_4

minCount{A_4}

C[i,j]는 A_i x … x A_j의 원소간 최소 곱셈 회수

C[1,4]

min{C[1,1] x C[2,4], C[1,2] x C[3,4], C[1,3] x C[4,4]}

C[1,3]

min{C[1,1] x C[2,3], C[1,2] x C[3,3]}

C[2,4]

min{C[2,2] x C[3,4], C[2,3] x C[4,4]}

C[1,2]

min{C[1,1] x C[2,2]}

C[2,3]

min{C[2,2] x C[3,3]}

C[3,4]

min{C[3,3] x C[4,4]}

C[1,1]

min{C[1,1]}

C[2,2]

min{C[2,2]}

C[3,3]

min{C[3,3]}

C[4,4]

min{C[4,4]}

부분 문제 풀이 순서

L = 3 **C[1,4]**

min{C[1,1] x C[2,4], C[1,2] x C[3,4], C[1,3] x C[4,4]}

L = 2 **C[1,3]** **C[2,4]**

min{C[1,1] x C[2,3], C[1,2] x C[3,3]} min{C[2,2] x C[3,4], C[2,3] x C[4,4]}

L = 1 **C[1,2]** **C[2,3]** **C[3,4]**

min{C[1,1] x C[2,2]} min{C[2,2] x C[3,3]} min{C[3,3] x C[4,4]}

L = 0 **C[1,1]** **C[2,2]** **C[3,3]** **C[4,4]**

min{C[1,1]} min{C[2,2]} min{C[3,3]} min{C[4,4]}

부분 문제를 일반화 (배열 표기)

- 일반적인 경우

C[1,4]

min{C[1,1] x C[2,4], C[1,2] x C[3,4], C[1,3] x C[4,4]}

- 규칙 도출

C[i, j]

min{ C[i, i] x C[i+1, j],
C[i, i+1] x C[i+2, j],
⋮
for k = i to j-1   C[i, k] x C[k+1, j],
⋮
C[i, j-2] x C[j-1, j],
C[i, j-1] x C[j, j] }

연속 행렬 곱셈 알고리즘

MatrixChain

입력: 연속된 행렬 $A_1 \times A_2 \times \cdots \times A_n$,

단, A_1은 $[d_0 \times d_1]$, A_2은 $[d_1 \times d_2]$, … , A_n은 $[d_{n-1} \times d_n]$

출력: 입력의 행렬 곱셈에 필요한 원소의 최소 곱셈 횟수

```
1.  for i = 1 to n                          // L=0
2.       C[i,i]= 0;
3.  for L = 1 to n−1 {                      // L은 부분 문제의 크기
4.       for i = 1 to n−L {                 // i는 부분 문제의 시작 위치
5.            j = i + L;                    // j는 부분 문제의 끝 위치
6.            C[i,j]= ∞;                    // C[i,j]는 부분문제 A_i × ⋯ × A_j의 해를 의미
7.            for k = i to j−1 {
8.                 temp = C[i,k]+d_{i−1}d_k d_j+C[k+1,j]
9.                 if (temp < C[i,j])
10.                     C[i,j]= temp
              }
       }
    }
11. return C[1,n]
```

MatrixChain 알고리즘 설명

- Line 1~2:
  - 배열의 대각선 원소들, $C[1,1], C[2,2], \cdots, C[n,n]$을 0으로 초기화
    - 배열의 대각선 원소들은 행렬 $A_1, A_2, \ldots, A_n$을 계산하는데 필요한 원소 간의 곱셈 횟수로서 실제로는 계산이 무의미하지만, 알고리즘상으로 크기가 1인 부분 문제 $C[i,i]$ 의 해를 뜻하므로 0을 설정한다.

C	1	2	3	…	n-1	n
1	0					
2		0				
3			0			
…				0		
n-1					0	
n						0

- Line 3~5:
  - Line 3: for 루프의 $L = 1\sim(n-1)$까지 변하는데,
    L은 부분 문제의 크기를 $2 \sim n$까지 증가시키는 사용되는 변수이다.
  - Line 4: for 루프의 $i = 1\sim(n-L)$까지 변하는데,
    i는 부분 문제의 시작 위치를 지정한다.
  - Line 5: $j = i + L$에서 j는 부분 문제의 끝 위치를 지정하여
    결국 $C[i,j]$는 $A_i \times \cdots \times A_j$의 원소간 최소 곱셈 회수를 의미한다.

- $L = 1$일 때, i는 $1 \sim (n-1)$, $j = i + 1$이 되어 크기가 2인 부분 문제 $(n-1)$개를 구한다.

- $L = 2$일 때, i는 $1 \sim (n-2)$, $j = i + 2$가 되어 크기가 3인 부분 문제 $(n-2)$개를 구한다.

$L = 1$
$\quad for\ i = 1 \sim (n-1)$
$\quad\quad j = i + 1$

C	1	2	3	…	n-1	n
1	0					
2		0				
3			0			
…				0		
n-1					0	
n						0

n-1개

$L = 2$
$\quad for\ i = 1 \sim (n-2)$
$\quad\quad j = i + 2$

C	1	2	3	…	n-1	n
1	0					
2		0				
3			0			
…				0		
n-1					0	
n						0

n-2개

- $L = n - 2$일 때, i는 $1 \sim (n-(n-2))$, $j = i + (n-2)$가 되어 크기가 $n-1$인 부분 문제 2개를 구한다.

- $L = n - 1$일 때, i는 $1 \sim (n-(n-1))$, $j = i + (n-1)$이 되어 크기가 n인 최종 문제 1개를 구한다.

$L = n - 2$
$\quad for\ i = 1 \sim 2$
$\quad\quad j = i + (n-2)$

C	1	2	3	…	n-1	n
1	0					
2		0				
3			0			
…				0		
n-1					0	
n						0

2개

$L = n - 1$
$\quad for\ i = 1 \sim 1$
$\quad\quad j = i + (n-1)$

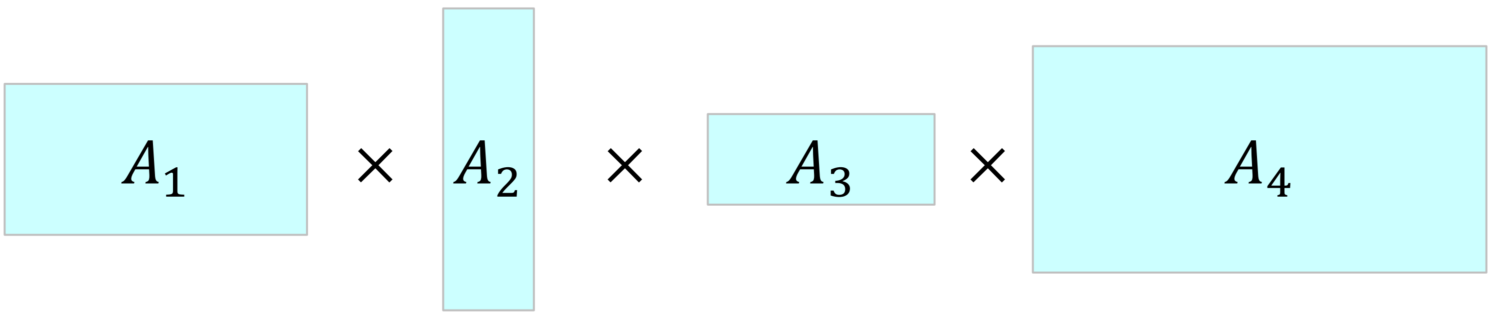
C	1	2	3	…	n-1	n
1	0					
2		0				
3			0			
…				0		
n-1					0	
n						0

1개

- Line 6~10:
  - Line 6: 최소 곱셈 횟수 $C[i,j]=\infty$로 초기화
  - Line 7~10: for루프 $k=i\sim(j-1)$까지 변하면서 모든 $(A_i\times\cdots\times A_k)\times(A_{k+1}\times\cdots\times A_j)$에 대해 원소간의 곱셈의 개수를 계산하고, 가장 작은 값을 $C[i,j]$에 저장한다.
  - 이때, 원소간의 곱셈의 개수는 $C[i,k]+d_{i-1}d_kd_j+C[k+1,j]$ 가 된다.
  - 여기서 부분 문제간의 함축적 순서가 존재함을 알 수 있다. 즉, $C[i,j]$의 해를 구하는데 부분 문제 $C[i,i],\ldots,C[i,j-1]$와 $C[i+1,j],\ldots,C[j,j]$의 해가 사용되는 관계가 확인된다.
- Line 11:
  - 주어진 문제의 해가 있는 $C[1,n]$을 리턴

MatrixChain 알고리즘의 수행 과정

- 다음 행렬의 곱 $A_1 \times A_2 \times A_3 \times A_4$에서 원소의 최소 곱셈 회수를 구하라.



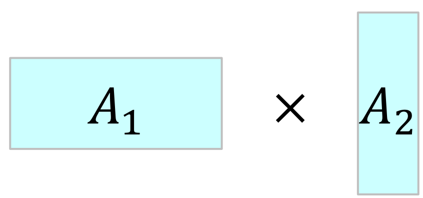
$[d_0 \times d_1]$ $= 10 \times 20$	$[d_1 \times d_2]$ $= 20 \times 5$	$[d_2 \times d_3]$ $= 5 \times 15$	$[d_3 \times d_4]$ $= 15 \times 30$

- Line 1~2:
  - $C[1,1] = C[2,2] = C[3,3] = C[4,4] = 0$으로 초기화

C	1	2	3	4
1	0			
2		0		
3			0	
4				0

- Line 3:
  - $L = 1\sim3$으로 증가하고,
    각각의 L값에 대하여 i가 변화하며 $C[i,j]$를 계산한다.

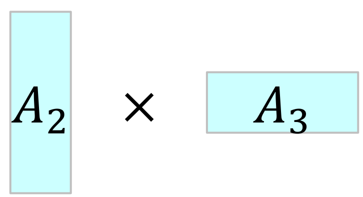
- $L = 1$일 때, $i = 1 \sim 3$까지 변한다.
  - $i = 1,\ j = 2$: 부분 문제 $A_1 \times A_2$ 의 해 $C[1,2]$를 구한다.
    먼저 $C[1,2] = \infty$로 초기화하고, $k = 1 \sim 1$까지 변한다.
    - $k = 1$일 때,
      $temp = C[1,1] + d_0 d_1 d_2 + C[2,2]$=0+(10×20×5)+0= 1,000
      현재 $C[1,2] = \infty$ 보다 $temp$가 작으므로 $C[1,2] = temp = 1{,}000$



$[d_0 \times d_1]$ =10 × 20 $[d_1 \times d_2]$ =20 × 5

- $i = 2,\ j = 3$: 부분 문제 $A_2 \times A_3$ 의 해 $C[2,3]$을 구한다.
  먼저 $C[2,3] = \infty$로 초기화하고, $k = 2$~2까지 변한다.

  - $k = 2$일 때,
    $temp = C[2,2] + d_1 d_2 d_3 + C[3,3]$=0+(20×5×15)+0= 1,500
    현재 $C[2,3] = \infty$ 보다 $temp$가 작으므로 $C[2,3] = temp = 1,500$



$[d_1 \times d_2]$ $=20 \times 5$　　$[d_2 \times d_3]$ $=5 \times 15$

- $i = 3,\ j = 4$: 부분 문제 $A_3 \times A_4$ 의 해 $C[3,4]$를 구한다.
  먼저 $C[3,4] = \infty$로 초기화하고, $k = 3\sim3$까지 변한다.
  - $k = 3$일 때,
    $temp = C[3,3] + d_2 d_3 d_4 + C[4,4]$=0+(5×15×30)+0=2,250
    현재 $C[3,4] = \infty$ 보다 $temp$가 작으므로 $C[3,4] = temp = 2,250$

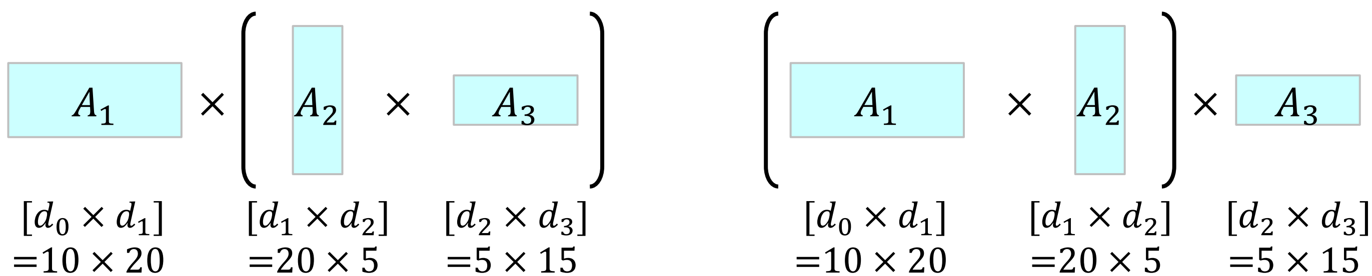
$A_3 \times A_4$

$[d_2 \times d_3]$ =5 × 15

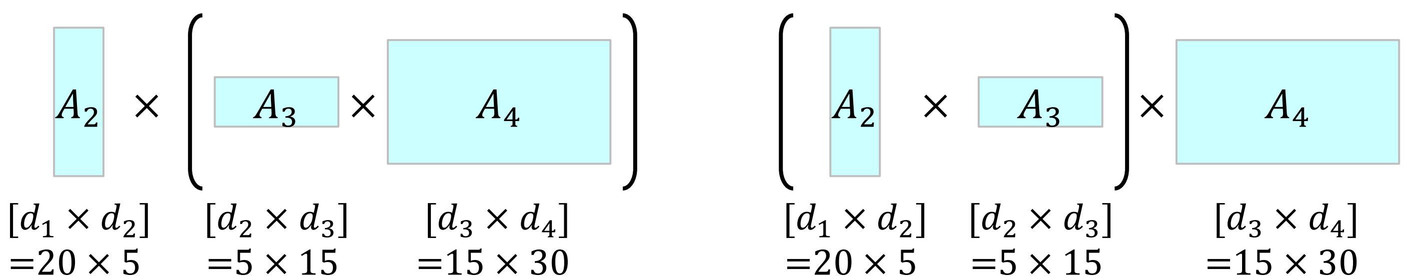
$[d_3 \times d_4]$ =15 × 30

C	1	2	3	4
1	0	1,000		
2		0	1,500	
3			0	2,250
4				0

- $L = 2$일 때, $i = 1\sim2$까지 변한다.
  - $i = 1,\ j = 3$: 부분 문제 $A_1 \times A_2 \times A_3$ 의 해 $C[1,3]$을 구한다.
    먼저 $C[1,3] = \infty$로 초기화하고, $k = 1\sim2$까지 변한다.
    - $k = 1$일 때,
      $temp = C[1,1] + d_0d_1d_3 + C[2,3]$=0+(10×20×15)+1,500=4,500
      현재 $C[1,3] = \infty$ 보다 $temp$가 작으므로 $C[1,3] = temp = 4{,}500$
    - $k = 2$일 때,
      $temp = C[1,2] + d_0d_2d_3 + C[3,3]$=1,000+(10×5×15)+0=1,750
      현재 $C[1,3] = 4{,}500$ 보다 $temp$가 작으므로 $C[1,3] = temp = 1{,}750$



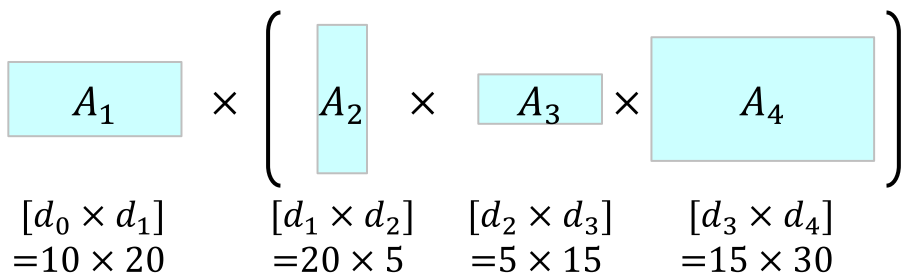
- $i = 2,\ j = 4$: 부분 문제 $A_2 \times A_3 \times A_4$ 의 해 $C[2,4]$를 구한다. 먼저 $C[2,4] = \infty$로 초기화하고, $k = 2{\sim}3$까지 변한다.
    - $k = 2$일 때,
      $temp = C[2,2] + d_1 d_2 d_4 + C[3,4] = 0 + (20 \times 5 \times 30) + 2{,}250 = 5{,}250$
      현재 $C[2,4] = \infty$ 보다 $temp$가 작으므로 $C[2,4] = temp = 5{,}250$
    - $k = 3$일 때,
      $temp = C[2,3] + d_1 d_3 d_4 + C[4,4] = 1{,}500 + (20 \times 15 \times 30) + 0 = 10{,}500$
      현재 $C[2,4] = 5{,}250$ 보다 $temp$가 크므로 $C[2,4] = 5{,}250$ 유지



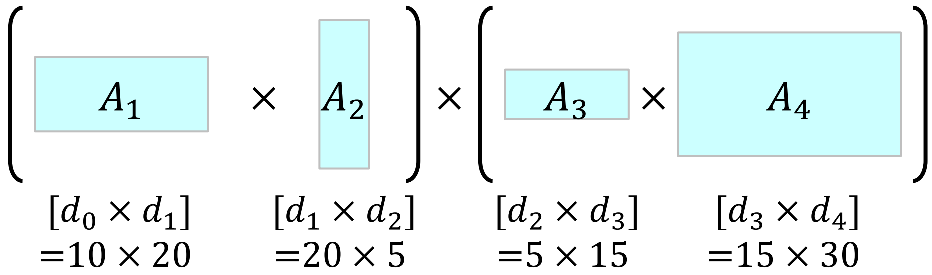
C	1	2	3	4
1	0	1,000	1,750	
2		0	1,500	5,250
3			0	2,250
4				0

부분 문제의 해 현황

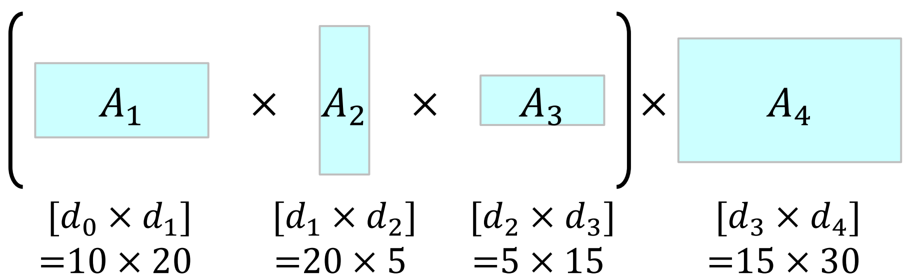
- $L = 3$일 때, $i = 1 \sim 1$까지 변한다.
  - $i = 1,\ j = 4$: 부분 문제 $A_1 \times A_2 \times A_3 \times A_4$ 의 해 $C[1,4]$를 구한다. 먼저 $C[1,4] = \infty$로 초기화하고, $k = 1 \sim 3$까지 변한다.
    - $k = 1$일 때,
      $temp = C[1,1] + d_0 d_1 d_4 + C[2,4]$=0+(10×20×30)+5,250=11,250
      현재 $C[1,4] = \infty$ 보다 $temp$가 작으므로 $C[1,4] = temp = 11,250$



◆ $k = 2$일 때,
$temp = C[1,2] + d_0d_2d_4 + C[3,4]$=1,000+(10×5×30)+2,250=4,750
현재 $C[1,4] = 11{,}250$ 보다 $temp$가 작으므로 $C[1,4] = temp = 4{,}750$



◆ $k = 3$일 때,
$temp = C[1,3] + d_0d_3d_4$+C[4,4]=1,750+(10×15×30)+0=6,250
현재 $C[1,4] = 4{,}750$ 보다 $temp$가 크므로 $C[1,4] = 4{,}750$ 유지



- MatrixChain 알고리즘 수행 결과
  - 원소의 최소 곱셈회수는 4,750이고,
    최소 곱셈회수가 나오는 연산 순서는 $(A_1 \times A_2) \times (A_3 \times A_4)$ 이다.

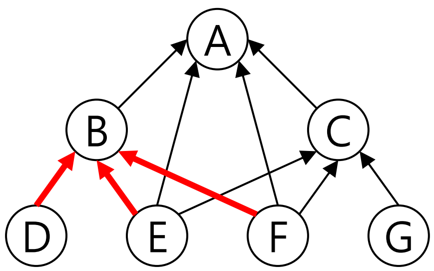
C	1	2	3	4
1	0	1,000	1,750	4,750
2		0	1,500	5,250
3			0	2,250
4				0

시간복잡도

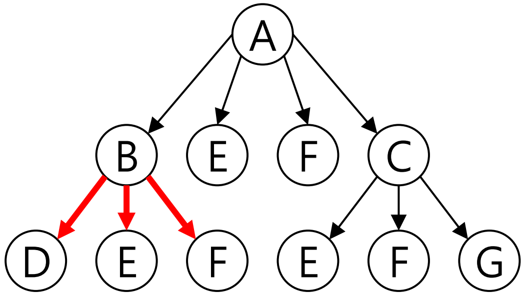
- MatrixChain 알고리즘의 시간복잡도
  - 배열 C가 $n \times n$이고, 원소의 수는 n^2인데,
    약 1/2 정도의 원소들의 값을 계산해야 한다.
  - 하나의 원소 $C[i,j]$를 계산하기 위해서는
    k −루프가 최대 $(n-1)$번 수행되어야 한다.
  - 따라서, MatrixChain 알고리즘의 시간복잡도는
    이다.

분할 정복과 비교

- 동적 계획 알고리즘을 분할 정복 알고리즘으로 구현하면, 부분 문제의 해를 중복 계산하는 문제 발생



동적 계획 알고리즘



분할 정복 알고리즘

Q&A