Dynamic Programming

Soontharee Koompairojn

30/3/15

Algorithmic Paradigms

- Greedy.
  - Build up a solution incrementally,
  - myopically optimizing some local criterion.

- Divide-and-conquer.
  - Break up a problem into sub-problems,
  - solve each sub-problem independently, and
  - combine solution to sub-problems to form solution to original problem.

- Dynamic programming.
  - Break up a problem into a series of overlapping sub-problems,
  - And build up solutions to larger and larger sub-problems.

Ref: Algorithm design, Jon Kleinberg and Eva Tardos, slide by Kevin Wayne

Dynamic Programming History

•Bellman. [1950s] Pioneered the systematic study of dynamic programming.

•Etymology.

- Dynamic programming = planning over time.
- Secretary of Defense was hostile to mathematical research.
- Bellman sought an impressive name to avoid

"it's impossible to use dynamic in a pejorative sense"
"something not even a Congressman could object to"

Reference: Bellman, R. E. *Eye of the Hurricane, An Autobiography.*

Ref: Algorithm design, Jon Kleinberg and Eva Tardos, slide by Kevin Wayne

Dynamic Programming Applications

- Areas.
  - Bioinformatics.
  - Control theory.
  - Information theory.
  - Operations research.
  - Computer science: theory, graphics, AI, compilers, systems, ….

- Some famous dynamic programming algorithms.
  - Unix diff for comparing two files.
  - Viterbi for hidden Markov models.
  - Smith-Waterman for genetic sequence alignment.
  - Bellman-Ford for shortest path routing in networks.
  - Cocke-Kasami-Younger for parsing context free grammars.

Ref: Algorithm design, Jon Kleinberg and Eva Tardos, slide by Kevin Wayne

Fibonacci Numbers

- Finding the n^{th} Fibonacci number F_n, where,
- $F_0 = 0$, $F_1 = 1$
- For all $i \geq 2$, $F_i = F_{i-1} + F_{i-2}$

- Recursive Algorithm

- Run time

Fibonacci Numbers

Finding the n^{th} Fibonacci number F_n, where,
$F_0 = 0$, $F_1 = 1$
For all $i \geq 2$, $F_i = F_{i-1} + F_{i-2}$

Dynamic Programming Algorithm

Run time

ปัญหา Maximum Subsequence Sum

กำหนดให้ S เป็นลำดับ (sequence) ตัวเลข n ตัว $\langle x_1, x_2, \ldots, x_n \rangle$,

โดยอาจจะมีจำนวนลบอยู่ด้วย

ปัญหา ให้คำนวณหา ผลบวกของลำดับย่อย subsequences ของ S ที่มีค่ามากที่สุด

กำหนดให้ s' = subsequence ที่ต่อเนื่องของ S

= $x_i, x_{i+1}, \ldots, x_{i+k}$, โดยที่ $1 \le i \le n$ และ $k \ge 0$

สมมติ $S = \langle 1, -5, 2, -1, 3 \rangle$,

ตัวอย่างของ S' ได้แก่

$\langle 1 \rangle$, $\langle 2, -1, 3 \rangle$,และ $\langle -5, 2 \rangle$

MSS ของ sequence S เป็นเท่าไร?

Maximum subsequence sum problem(2)

Developing a dynamic programming algorithm

- กำหนดโครงสร้างของ optimal solution

  แบ่งปัญหาออกเป็นปัญหาย่อย

แนวคิด : เก็บผลรวมที่มากที่สุดของsubsequenceใดๆ สิ้นสุดที่ตัวเลข x_i, โดยที่ $1 \le i \le n$

กำหนดให้ B เป็น array 1 มิติขนาด n ตัว

B[i] เก็บค่า ผลบวกที่มากที่สุดของ subsequence สิ้นสุดที่ตัวเลข x_i

เมื่อคำนวณค่าในตาราง B จะได้ค่าผลบวกที่มากที่สุดเอง

Pseudocode

$B[1] = x_1$

For i = 1 to n do

$$B[i+1] = \begin{cases} B[i] + x_{i+1} & \text{if } B[i] > 0 \\ x_{i+1} & \text{if } B[i] \le 0 \end{cases}$$

- สมมติ $S = \langle 1, -5, 2, -1, 3 \rangle$
- MSS ของ sequence S เป็นเท่าไร?
- ให้เติมค่าในตาราง B ที่มีขนาด 5 ตัว

Subset sum problem

กำหนดให้ เซต $S = \{a_1, a_2, \ldots, a_n\}$ เป็นเซตของเลขจำนวนเต็มบวก

และค่าผลรวมเป็นเลขจำนวนเต็ม B

คำถาม: มีเซตย่อย (subset) S' ของ S ที่ผลบวกของสมาชิกใน S'

มีค่าเท่ากับ B หรือไม่

ตัวอย่างเช่น กำหนดให้ เซต $S = \{3, 34, 4, 12, 5, 2\}$,

และค่าผลรวมเท่ากับ 9

คำถาม: มีเซตย่อย (subset) S' ของ S ที่ผลบวกของสมาชิกใน S'

มีค่าเท่ากับ 9 หรือไม่

มี หรือ ไม่มี , True or False ?

Idea การแก้ปัญหา subset sum

Given: A (multi-)set S of $\{a_1, a_2, \ldots, a_n\}$ of positive integers and a positive integer B

Question: Is there a subset S' of S such that the sum of the elements in S' is equal to B?

- กำหนดคำถามในรูปฟังก์ชัน T(n,B)

  โดยที่ n คือ จำนวนตัวเลข

  B เป็น ตัวเลขผลรวมตามโจทย์กำหนด

- แตกเป็นปัญหาย่อย

จากแนวคิดที่ ค่าผลรวมตัวหลัง ๆ มาจาก เซตของตัวเลขในช่วงแรก ๆ

กำหนดให้ T(i,j) เป็นคำถามว่า มี subset S' ใน $\{a_1, a_2, \ldots, a_i\}$ ที่มีผลบวกเท่ากับ j หรือไม่ ?

การพิสูจน์ โดยวิธี Induction

พิสูจน์ว่า T(n,B) เป็นจริงหรือไม่
(พิสูจน์ว่ามีเซตย่อยจากจำนวน n ตัวที่มีผลบวก= B หรือไม่)

- Base case:

  ถ้าตัวเลขตัวใดตัวหนึ่งมีขนาด B แล้วจะเป็นจริง

- Induction step (วิเคราะห์ครั้งที่ 1):
  - ถ้าสำหรับฟังก์ชัน T(n-1, B) เป็น**จริง,**
    นั่นคือเราไม่จำเป็นต้องใช้ $\mathbf{a_n}$
  - ถ้าสำหรับฟังก์ชัน T(n-1, B) เป็น**เท็จ,**
    นั่นคือ เราต้องใช้ $\mathbf{a_n}$
    กรณีนี้ เซตของตัวเลขก่อนหน้านี้จะมีค่าผลรวมเป็น $B- a_n$
  - ดังนั้นเราแบ่ง ปัญหา T(n,B) ออกเป็น
    ปัญหาย่อย T(n-1,B) และ $T(n-1, B- a_n)$

การพิสูจน์ โดยวิธี Induction(2)

- เนื่องจากเราจะต้องมีการแบ่งเป็นปัญหาย่อย ๆ

  นั่นคือ การจะแก้ปัญหา subset sum ที่มีผลรวมเท่ากับ B

  เราต้องแก้ปัญหา subset sum ที่มีค่าน้อยกว่าหรือเท่ากับ B ด้วย

- Induction step (วิเคราะห์ครั้งที่ 2):

  สำหรับเซตตัวเลข i ตัว $\{a_1, a_2, \ldots, a_i\}$

  ให้ j เป็นผลบวกของเซตตัวเลข i ตัว

  - ถ้า $j = 0$ นั่นคือ ไม่มีตัวเลขตัวไหนอยู่ในเซตคำตอบ
  - ถ้า $a_1 = j$ นั่นคือ a_1 เป็นสมาชิกตัวเดียวในเซตคำตอบ

วิเคราะห์ปัญหาร่วมกับ table

		ผลรวม j															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
first item i	i=1	0																
	i=2	0																
	i=3	0																
	i=4	0																

การพิสูจน์ โดยวิธี Induction(2)

- แบ่งปัญหา $T(i,j)$ ออกเป็น
  ปัญหาย่อย $T(i-1, j)$ และ $T(i-1, j- a_i)$
  สมมติเรารู้คำตอบของ $T(i-1,j)$ โดยที่ $0 \le j \le B$,
  ถ้า $j- a_i < 0$ เราก็ไม่ต้องสนใจปัญหาย่อยนี้

Subset sum problem(2)

Developing a dynamic programming algorithm

- กำหนดโครงสร้างของ optimal solution

  แบ่งปัญหาออกเป็นปัญหาย่อย

แนวคิด : เราเก็บคำตอบของปัญหาย่อยไว้ใน 2D Boolean array T,

ที่มีขนาด n แถว และ B+1 คอลัมน์.

ถ้าเซตย่อย (subset) $\{a_1, a_2, \ldots, a_i\}$ มีผลรวมเท่ากับ j

ให้ค่า T[i,j] = true

มิฉะนั้น ให้ค่า T[i,j] = false

Subset sum problem(3)

สำหรับแถวแรก,

for $0 \le j \le B$

  if (j=0) or ($a_1 = j$)

    T[1,j] = true

  else T[1,j] = false

สำหรับแถว **i**,

for $0 \le j \le B$

  if ((T[i-1,j]==true) or (T[i-1,j- a_i] ==true))

    T[i,j] = true

  else T[i,j] = false

ตัวอย่าง

เซต S = <2,3,5,6>

ผลรวม B = 16

Is there a subset (มี Subsetหรือไม่)?

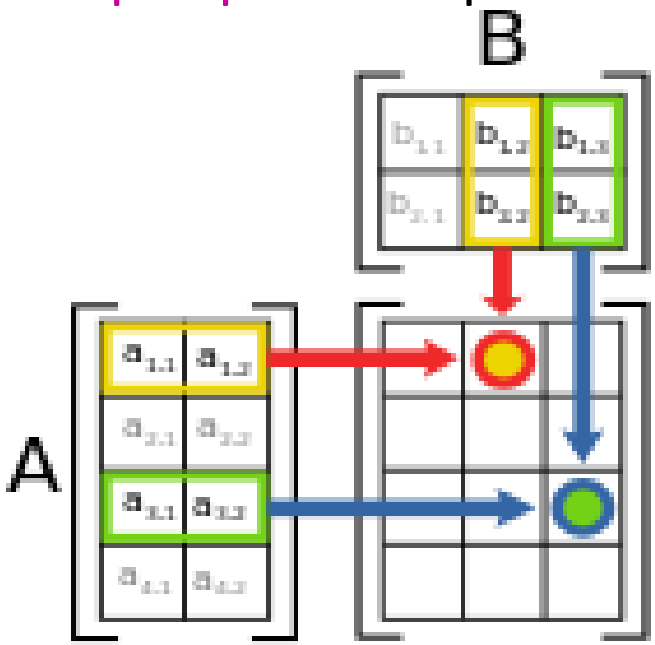
True or False?

	ผลรวม B															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a_1=2																	
a_2=3																	
a_3=5																	
a_4=6																	

Matrix Multiplication

- Let A be a p x q matrix
- Let B be a q x r matrix

- Let C be the matrix product AB ,a p x r matrix
- It uses p x q x r multiplications.



$$c[i,j] = \sum_{k=1}^{q} a[i,k]b[k,j]$$

For $1 \le i \le p$ and $1 \le j \le r$

http://en.wikipedia.org/wiki/Matrix_multiplication

Matrix Multiplication (1)

- Matrix multiplication is associative.
- For matrices A, B and C, A(BC) = (AB)C
- Suppose A is a r x 2 matrix
  B is a 2 x r matrix
  C is a r x 1 matrix
- How many multiplication of the product A(BC) ?
- How many multiplication of the product (AB)C ?

Matrix Chain Multiplication

Given : A chain $\langle A_1 A_2 \ldots A_n \rangle$ of matrices
where A_i is a $p_{i-1} \times p_i$ matrix $(1 \le i \le n)$

Required : Fully parenthesize the product $A_1 A_2 \ldots A_n$
so that the number of scalar multiplications
is minimized.

Method

1. Exhaustive search?
   the run time is exponential in n

   that is , $\binom{2n}{n} / (n+1) = \Omega(4^n / n^{3/2})$

2. Dynamic Programming

Matrix Chain Multiplication(2)

- Developing a dynamic programming algorithm

- Determine the structure of an optimal solution
  Decompose the problem into subproblems.

Let $M = A_1 A_2 \ldots A_n = M_1 M_2$
where $M_1 = A_1 A_2 \ldots A_k$
$M_2 = A_{k+1} A_{k+2} \ldots A_n$

Observation: Regardless of the value of **k**,
for the evaluation of M to be optimal,
the evaluations of M_1 and M_2 must themselves
be solved optimally.

Matrix Chain Multiplication(3)

For $1 \le i \le j \le n$

Let m[i,j] stores the minimum cost
(the minimum number of multiplications used)
for computing the subchain $A_i A_{i+1} \ldots A_j$

Let A_i be a $p_{i-1} \times p_i$ matrix $(1 \le i \le n)$

For all the possible values k such that $i \le k < j$
Where the product $A_i A_{i+1} \ldots A_j$

From the definition of m , m[i,j] = 0 where i=j
To evaluate the product optimally, split at k,
We incur the following 3 costs:
Cost m[i,k] evaluating $A_i A_{i+1} \ldots A_k$
Cost m[k+1,j] evaluating $A_{k+1} A_{k+2} \ldots A_j$
Cost $p_{i-1} p_k p_j$ evaluating the product of 2 matrices

Therefore,

$$m[i,j] = \min_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1} p_k p_j]$$

Matrix Chain Multiplication(4)

Example : Given a chain of 4 matrices

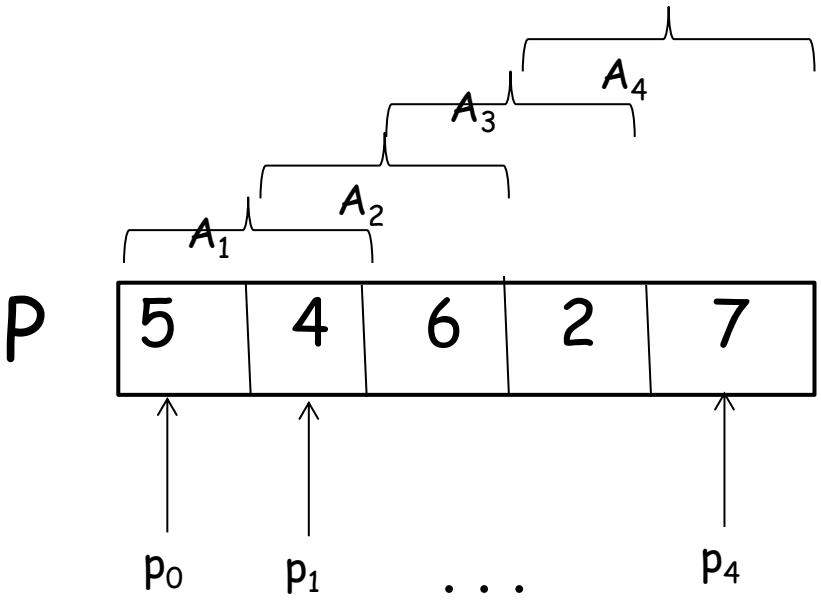
Let A_1 be a 5x4 matrix

A_2 be a 4x6 matrix

A_3 be a 6x2 matrix

A_4 be a 2x7 matrix

Find m[1,4] = ?



m[i,j]	1	2	3	4
1	0			
2		0		
3			0	
4				0

Matrix Chain Multiplication(5)

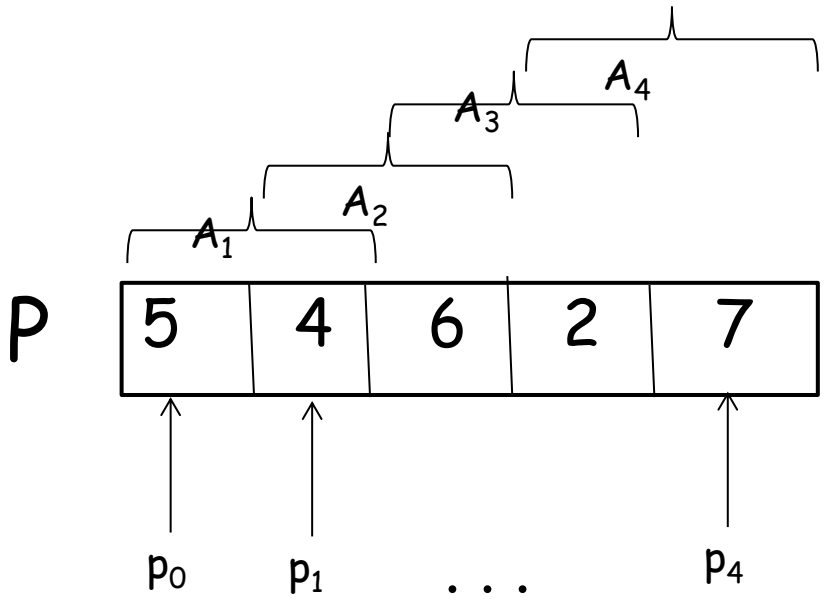
Step1 : Computing m[1,2]

definition $m[i,j] = \min_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1}\, p_k\, p_j]$

$m[1,2] = \min_{1 \le k < 2} [m[1,k] + m[k,2] + p_0\, p_k\, p_2]$

$= m[1,1] + m[2,2] + p_0\, p_1\, p_2$

$= 0 + 0 + 5 \times 4 \times 6 = 120$



m[i,j]	1	2	3	4
1	0	120		
2		0		
3			0	
4				0

Matrix Chain Multiplication(6)

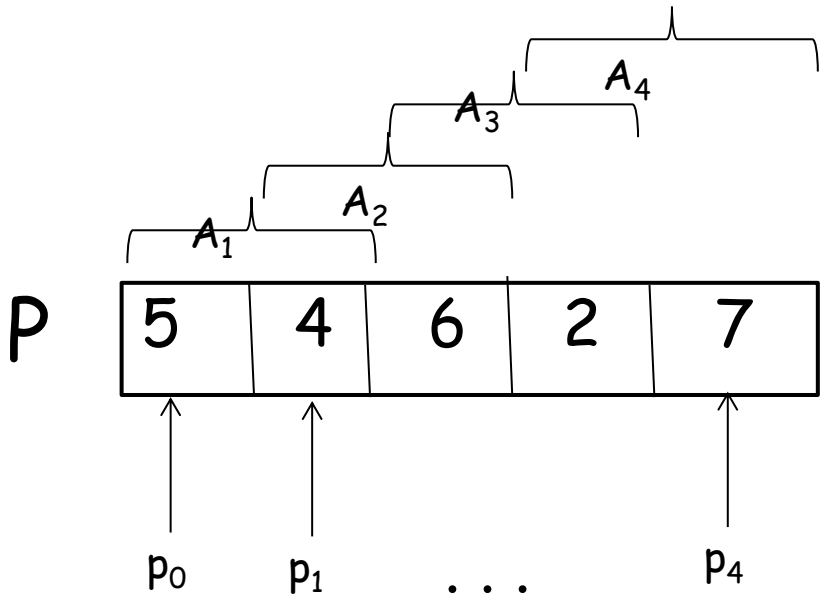
Step 2: Computing m[2,3]

definition $m[i,j] = \min_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1}\, p_k\, p_j]$

$m[2,3] = \min_{2 \le k < 3} [m[2,k] + m[k,3] + p_1\, p_k\, p_3]$

$= m[2,2] + m[3,3] + p_1\, p_2\, p_3$

$= 0 + 0 + 4 \times 6 \times 2 = 48$



m[i,j]	1	2	3	4
1	0	120		
2		0	48	
3			0	
4				0

Matrix Chain Multiplication(7)

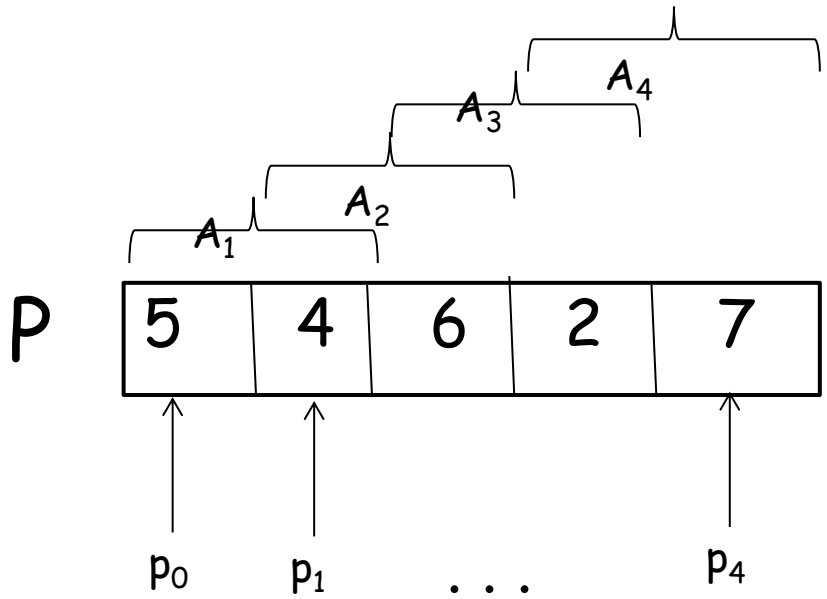
Step 3: Computing m[3,4]

definition m[i,j] = $\min_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1}\, p_k\, p_j]$

m[3,4] =

=

=



m[i,j]	1	2	3	4
1	0	120		
2		0	48	
3			0	
4				0

Matrix Chain Multiplication(8)

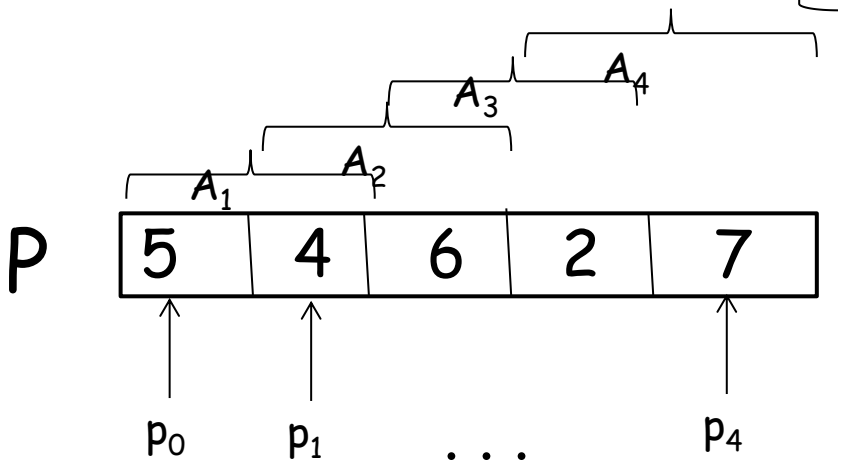
Step 4: Computing m[1,3]

definition m[i,j] = $\min_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1}\, p_k\, p_j]$

$$m[1,3] = \min_{1 \le k < 3} [m[1,k] + m[k+1,3] + p_0 p_k p_3]$$

$$m[1,3] = \min \begin{cases} m[1,1]+m[2,3]+ p_0p_1p_3 \\ m[1,2]+m[3,3]+ p_0p_2p_3 \end{cases}$$

$$= \min \begin{cases} 0 + 48 + 5\times4\times2 \\ 120 + 0 + 5\times6\times2 \end{cases} = 88$$



m[i,j]	1	2	3	4
1	0	120	88	
2		0	48	
3			0	84
4				0

Matrix Chain Multiplication(9)

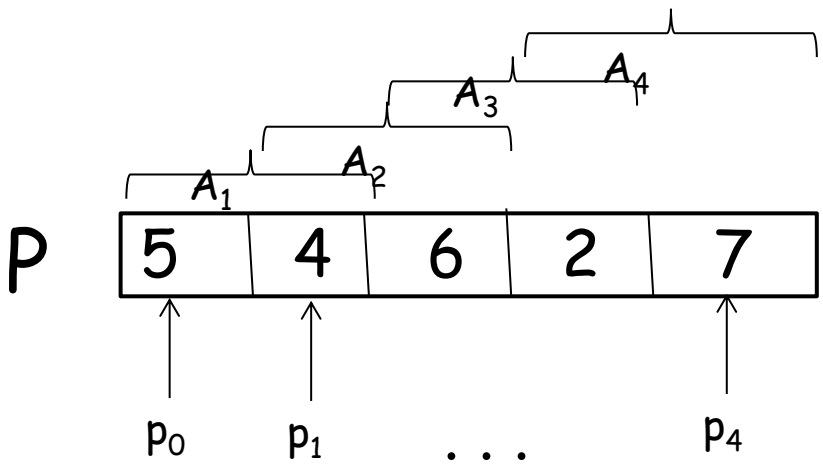
Step 5: Computing m[2,4]

definition m[i,j] = $\min_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1}\, p_k\, p_j]$

$m[2,4] =$

=

=



m[i,j]	1	2	3	4
1	0	120	88	
2		0	48	
3			0	84
4				0

Matrix Chain Multiplication(10)

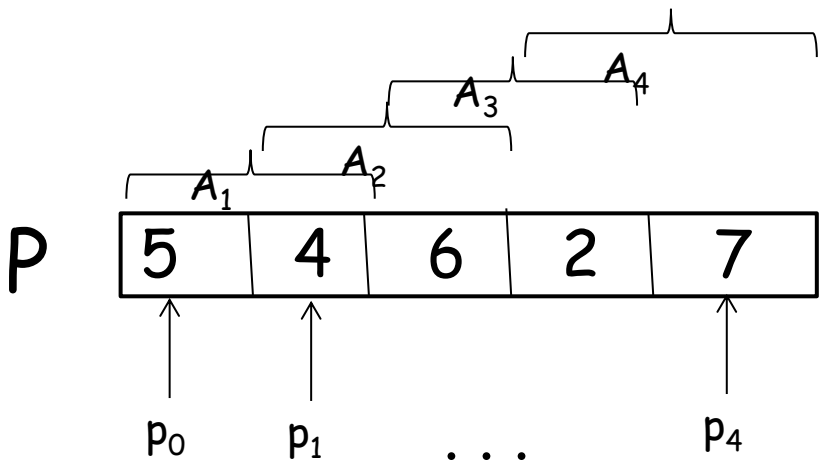
Step 6: Computing m[1,4]

definition m[i,j] = $\min_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1}\, p_k\, p_j]$

$$m[1,4] = \min_{1 \le k < 4} [m[1,k] + m[k+1,4] + p_0 p_k p_4]$$

$$= \min \begin{cases} m[1,1]+m[2,4]+ p_0 p_1 p_4 \\ m[1,2]+m[3,4]+ p_0 p_2 p_4 \\ m[1,3]+m[4,4]+ p_0 p_3 p_4 \end{cases}$$

$=$



m[i,j]	1	2	3	4
1	0	120	88	
2		0	48	104
3			0	84
4				0

Matrix Chain Multiplication(11)

Pseudocode

1. for i = 1 to n do
   m[i,i] = 0; // subscript diffeence = 0

2. for q = 1 to n-1 do
   // q represents the subscript difference

   2.1 for i = 1 to n-q do
   2.1.1 j = i + q;
   2.1.2 m[i,j] = minimum over i <= k < j of
   $[m[i,k] + m[k+1,j] + p_{i-1}p_k p_j]$;

3. Print m[1,n] // optimal solution