Dynamic Programming

Presentation by
V. Balasubramanian
SSN College of Engineering

Dynamic Programming

- Dynamic Programming is a general algorithm design technique for solving problems defined by recurrences with overlapping subproblems
- Invented by American mathematician Richard Bellman in the 1950s to solve optimization problems and later assimilated by CS

- The term "dynamic programming" comes from control theory, and
- in this sense "programming" means the use of an array (table) in which a solution is constructed.

Dynamic Programming

- “Programming” here means “planning”
- Main idea:
  - set up a recurrence relating a solution to a larger instance to solutions of some smaller instances
- - solve smaller instances once
  - record solutions in a table
  - extract solution to the initial instance from that table

Permutations

- Suppose we have four balls marked A, B, C, and D in an urn or container, and two balls will be drawn
- AB & BA are different. 4(3) = 12

AB	AC	AD
BA	BC	BD
CA	CB	CD
DA	DB	DC.

Strategy

The steps in the development of a dynamic programming algorithm are as follows:

1. *Establish* a recursive property that gives the solution to an instance of the problem.
2. Solve an instance of the problem in a *bottom-up* fashion by solving smaller instances first.

Contd…

- if we have four balls and three are
- drawn, the first ball can be any one of four; once the first ball is drawn, the second ball can be any of three; and once the second ball is drawn, the third ball can be any of two.
- 4.3.2 = 24.

Contd…

- In general, if we have n balls, and we are picking k of them,
- $(n)(n - 1) \bullet \bullet \bullet (n - k + 1)$.
- If $k=n$,
  - $(n)(n - 1) \bullet \bullet \bullet (n - n + 1) = n!$.
-

Combinations

- A and B, A and C, A and D, B and C,B and D, C and D.
- 6 distinct outcomes.
- 4(3)/2 =6.
- 3 balls to be taken.
- 4(3)(2)/3! = 4.

Contd…

- if there are n balls and k balls are drawn, then

$$\frac{(n)(n-1)\cdots(n-k+1)}{k!}.$$

$$(n)(n-1)\cdots(n-k+1) = (n)(n-1)\cdots(n-k+1) \times \frac{(n-k)!}{(n-k)!}$$

$$= \frac{n!}{(n-k)!},$$

Contd…

- Binomial coefficients are coefficients of the binomial formula:
- $(a + b)^n = C(n,0)a^nb^0 + \ldots + C(n,k)a^{n-k}b^k + \ldots + C(n,n)a^0b^n$
- Recurrence:
- $C(n,k) = C(n-1,k) + C(n-1,k-1)$ for $n > k > 0$
- $C(n,0) = 1$, $C(n,n) = 1$ for $n >= 0$

$$(a+b)^n = \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} a^k b^{n-k}.$$

The **binomial theorem** provides a useful method for raising any binomial to a nonnegative integral power.

Consider the patterns formed by expanding $(x + y)^n$.

$(x + y)^0 = 1 \longleftarrow$ 1 term

$(x + y)^1 = x + y \longleftarrow$ 2 terms

$(x + y)^2 = x^2 + 2xy + y^2 \longleftarrow$ 3 terms

$(x + y)^3 = x^3 + 3x^2y + 3xy^2 + y^3 \longleftarrow$ 4 terms

$(x + y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4 \longleftarrow$ 5 terms

$(x + y)^5 = x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5 \leftarrow$ 6 terms

Notice that each expansion has $n + 1$ terms.

Example: $(x + y)^{10}$ will have 10 + 1, or 11 terms.

Consider the patterns formed by expanding $(x + y)^n$.

$$(x + y)^0 = 1$$

$$(x + y)^1 = x + y$$

$$(x + y)^2 = x^2 + 2xy + y^2$$

$$(x + y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$$

$$(x + y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4$$

$$(x + y)^5 = x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5$$

1. The exponents on x decrease from n to 0.
   The exponents on y increase from 0 to n.

2. Each term is of degree n.
   Example: The 5th term of $(x + y)^{10}$ is a term with x^6y^4."

The coefficients of the binomial expansion are called **binomial coefficients**. The coefficients have symmetry.

$$(x + y)^5 = 1x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + 1y^5$$

The first and last coefficients are 1.

The coefficients of the second and second to last terms are equal to n.

Example: What are the last 2 terms of $(x + y)^{10}$? Since $n = 10$, the last two terms are $10xy^9 + 1y^{10}$.

The coefficient of $x^{n-r}y^r$ in the expansion of $(x + y)^n$ is written $\binom{n}{r}$ or ${}_nC_r$. So, the last two terms of $(x + y)^{10}$ can be expressed as ${}_{10}C_9\, xy^9 + {}_{10}C_{10}\, y^{10}$ or as $\binom{10}{9}xy^9 + \binom{10}{10}y^{10}$.

The triangular arrangement of numbers below is called **Pascal's Triangle**.

$$
\begin{array}{lcccccccccccl}
 & & & & & & 1 & & & & & & 0^{th} \text{ row} \\
 & & & & & 1 & & 1 & & & & & 1^{st} \text{ row} \\
1 + 2 = 3 & & & & \boxed{1} & & \boxed{2} & & 1 & & & & 2^{nd} \text{ row} \\
 & & & 1 & & \boxed{3} & & 3 & & 1 & & & 3^{rd} \text{ row} \\
6 + 4 = 10 & & 1 & & 4 & & \boxed{6} & & \boxed{4} & & 1 & & 4^{th} \text{ row} \\
 & 1 & & 5 & & 10 & & \boxed{10} & & 5 & & 1 & 5^{th} \text{ row}
\end{array}
$$

Each number in the interior of the triangle is the sum of the two numbers immediately above it.

The numbers in the n^{th} row of Pascal's Triangle are the binomial coefficients for $(x + y)^n$.

Contd…

Theorem 1. *The binomial coefficients $B(n,k)$ defined by formula (1) satisfy*

$$(2) \qquad B(n,k) = B(n-1,k-1) + B(n-1,k), \qquad 1 \le k \le n-1.$$

Together with the initial conditions $B(n,0) = B(n,n) = 1$ recursion (2) completely specifies the binomial coefficients.

1

Proof

Proof. Using equation (1) we obtain

$$B(n-1,k-1) = \frac{(n-1)!}{(k-1)!(n-k)!} = \frac{(n-1)!}{(k-1)!(n-k-1)!}\frac{1}{n-k},$$

$$B(n-1,k) = \frac{(n-1)!}{k!(n-k-1)!} = \frac{(n-1)!}{(k-1)!(n-k-1)!}\frac{1}{k}.$$

Proof

$$\begin{aligned}
B(n-1,k-1)+B(n-1,k) &= \frac{(n-1)!}{(k-1)!(n-k-1)!}\left(\frac{1}{n-k}+\frac{1}{k}\right) \\
&= \frac{(n-1)!}{(k-1)!(n-k-1)!}\frac{n}{k(n-k)} \\
&= \frac{n!}{k!(n-k)!} \\
&= B(n,k).
\end{aligned}$$

	0	1	2	…	$k-1$	k
0	1					
1	1	1				
2	1	2	1			
⋮						
k	1					1
⋮						
$n-1$	1				$C(n-1, k-1)$	$C(n-1, k)$
n	1					$C(n, k)$

FIGURE 8.1 Table for computing the binomial coefficient $C(n, k)$ by the dynamic programming algorithm

n								
0	1							
1	1	1						
2	1	2	1					
3	1	3	3	1				
4	1	4	6	4	1			
5	1	5	10	10	5	1		
6	1	6	15	20	15	6	1	
7	1	7	21	35	35	21	7	1

Table: Pascal's triangle.

Algorithm

ALGORITHM *Binomial*(n, k)

//Computes $C(n, k)$ by the dynamic programming algorithm
//Input: A pair of nonnegative integers $n \geq k \geq 0$
//Output: The value of $C(n, k)$

for $i \leftarrow 0$ **to** n **do**
    for $j \leftarrow 0$ **to** $\min(i, k)$ **do**
        if $j = 0$ **or** $j = i$
            $C[i, j] \leftarrow 1$
        else $C[i, j] \leftarrow C[i-1, j-1] + C[i-1, j]$
return $C[n, k]$

Analysis Addition is the operation

$$A(n,k) = \sum_{i=1}^{k}\sum_{j=1}^{i-1} 1 + \sum_{i=k+1}^{n}\sum_{j=1}^{k} 1 = \sum_{i=1}^{k}(i-1) + \sum_{i=k+1}^{n} k$$

$$= \frac{(k-1)k}{2} + k(n-k) \in \Theta(nk).$$

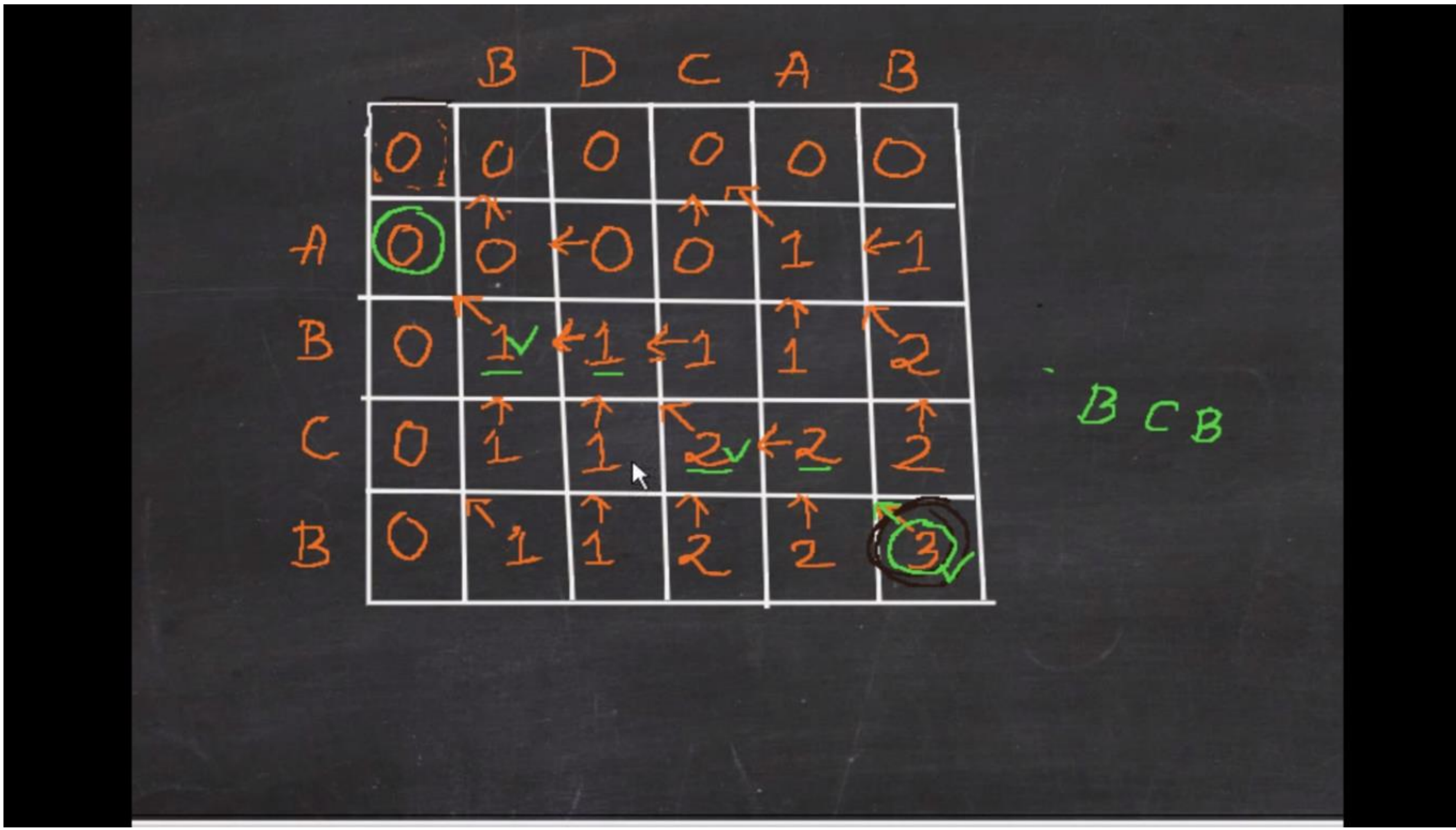
C(12,5)

	k					
n	0	1	2	3	4	5
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1
6	1	6	15	20	15	6
7	1	7	21	35	35	21
8	1	8	28	56	70	56
9	1	9	36	84	126	126
10	1	10	45	120	210	252
11	1	11	55	165	330	462
12	1	12	66	220	495	792

Use Excel for DEMO

Fact(12)	479001600
Fact(5)	120
Fact(7)	5040
12c5	792

Longest Common Subsequence


B D C A B
0 0 0 0 0 0
A 0 0 0 0 1 1
B 0 1 1 1 1 2
C 0 1 1 2 2 2
B 0 1 1 2 2 3
B C B

Dynamic Programming: Longest Common Subsequence [LCS]

Example

$X = \langle A, B, C, B, D, A \rangle$
$Y = \langle B, D, C, A, B, A \rangle$

$$c[i, j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1, j-1] + 1 & \text{if } x_i = y_j \\ \max(c[i, j-1], c[i-1, j]) & \text{if } x_i \neq y_j \end{cases}$$

If $x_i = y_j$
  $b[i, j] = $ " $\nwarrow$ "
Else if
  $c[i-1, j] \geq c[i, j-1]$
    $b[i, j] = $ " $\uparrow$ "
else
    $b[i, j] = $ " $\leftarrow$ "

		0	1	2	3	4	5	6
		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

add notes

10:37 / 11:59

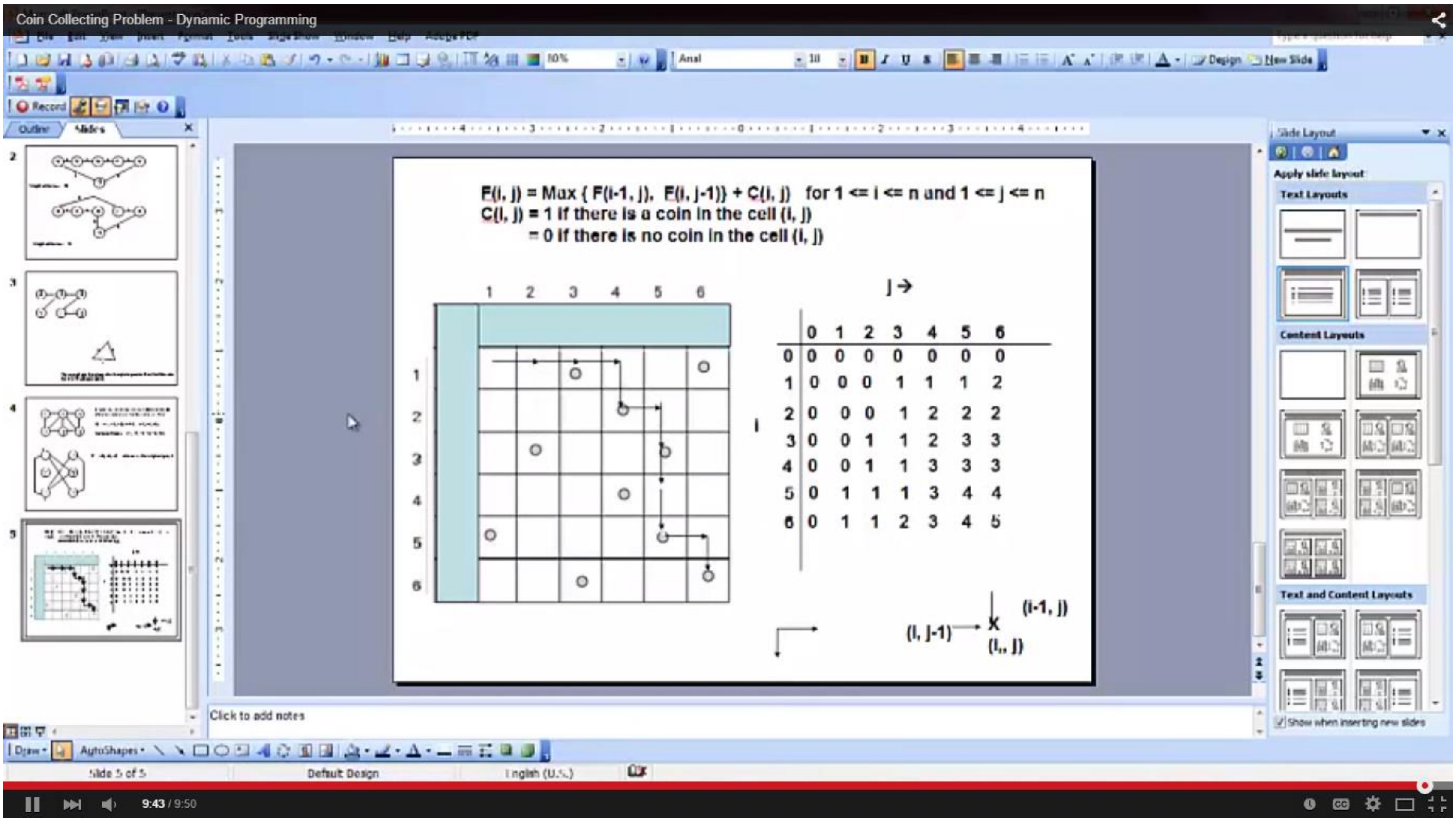
- C(I,j) = max (c(i-1,j), c(I,j-1) + 1 /0

Gift collection

	0	**1**	**2**	**3**	**4**	**5**	**6**
0							
1				Gift			gift
2					Gift		
3			Gift			Gift	
4					Gift		
5		Gift				Gift	
6				Gift			Gift

Gift collection

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	2
2	0	0	0	1	2	2	2
3	0	0	1	1	2	3	3
4	0	0	1	1	3	3	3
5	0	1	1	1	3	4	4
6	0	1	1	2	3	4	5


Coin Collecting Problem - Dynamic Programming
F(i, j) = Max { F(i-1, j), F(i, j-1)} + C(i, j) for 1 <= i <= n and 1 <= j <= n
C(i, j) = 1 if there is a coin in the cell (i, j)
= 0 if there is no coin in the cell (i, j)
j →
i
0 1 2 3 4 5 6
0 0 0 0 0 0 0 0
1 0 0 0 1 1 1 2
2 0 0 0 1 2 2 2
3 0 0 1 1 2 3 3
4 0 0 1 1 3 3 3
5 0 1 1 1 3 4 4
6 0 1 1 2 3 4 5
(i-1, j)
(i, j-1)
(i, j)
Click to add notes
Slide 5 of 5
Default Design
9:43 / 9:50

Matrix Chain Multiplication

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 & 9 & 1 \\ 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 29 & 35 & 41 & 38 \\ 74 & 89 & 104 & 83 \end{bmatrix}$$

In general, to multiply an $i \times j$ matrix times a $j \times k$ matrix, using the standard method, it is necessary to do

$$i \times j \times k \text{ elementary multiplications.}$$

Example

$$\begin{array}{ccccccc} A & \times & B & \times & C & \times & D \\ 20 \times 2 & & 2 \times 30 & & 30 \times 12 & & 12 \times 8 \end{array}$$

5 different orders

$A(B(CD))$	$30 \times 12 \times 8 + 2 \times 30 \times 8 + 20 \times 2 \times 8 = 3{,}680$
$(AB)(CD)$	$20 \times 2 \times 30 + 30 \times 12 \times 8 + 20 \times 30 \times 8 = 8{,}880$
$A((BC)D)$	$2 \times 30 \times 12 + 2 \times 12 \times 8 + 20 \times 2 \times 8 = 1{,}232$
$((AB)C)D$	$20 \times 2 \times 30 + 20 \times 30 \times 12 + 20 \times 12 \times 8 = 10{,}320$
$(A(BC))D$	$2 \times 30 \times 12 + 20 \times 2 \times 12 + 20 \times 12 \times 8 = 3{,}120$

$$A_1\underbrace{(A_2A_3 \cdots A_n)}_{t_{n-1}\ \text{different orders}}$$

$$t_n \geq 2^{n-2}.$$

$$\begin{array}{ccccccccccc} A_1 & \times & A_2 & \times & A_3 & \times & A_4 & \times & A_5 & \times & A_6 \\ 5 \times 2 & & 2 \times 3 & & 3 \times 4 & & 4 \times 6 & & 6 \times 7 & & 7 \times 8 \\ d_0 \quad d_1 & & d_1 \quad d_2 & & d_2 \quad d_3 & & d_3 \quad d_4 & & d_4 \quad d_5 & & d_5 \quad d_6 \end{array}$$

$$M[1][6] = \underset{1 \le k \le 5}{minimum}(M[1][k] + M[k+1][6] + d_0 d_k d_6).$$

Recurrence equation

$$M[i][j] = \underset{i \le k \le j-1}{minimum}(M[i][k] + M[k+1][j] + d_{i-1}d_k d_j) \qquad \text{if } i < j$$

$$M[i][i] = 0.$$

Compute diagonal 0:

$$M[i][i] = 0 \qquad \text{for } 1 \le i \le 6.$$

$$\begin{aligned} M[1][2] &= \underset{1 \le k \le 1}{minimum}(M[1][k] + M[k+1][2] + d_0 d_k d_2) \\ &= M[1][1] + M[2][2] + d_0 d_1 d_2 \\ &= 0 + 0 + 5 \times 2 \times 3 = 30 \end{aligned}$$

Compute diagonal 2:

$$\begin{aligned} M[1][3] &= \underset{1 \le k \le 2}{minimum}(M[1][k] + M[k+1][3] + d_0 d_k d_3) \\ &= minimum(M[1][1] + M[2][3] + d_0 d_1 d_3, \\ &\qquad\qquad M[1][2] + M[3][3] + d_0 d_2 d_3) \\ &= minimum(0 + 24 + 5 \times 2 \times 4, 30 + 0 + 5 \times 3 \times 4) = 64 \end{aligned}$$

	1	2	3	4	5	6
1	0	30	64	132	226	348
2		0	24	72	156	268
3			0	72	198	366
4				0	168	392
5					0	336
6						0

Final answer

P(1,6)=1, P(2,6)=5

$$A_1(A_2A_3A_4A_5A_6).$$

$$(A_2A_3A_4A_5)A_6.$$

$$A_1((((A_2A_3)A_4)A_5)A_6).$$

P Matrix

	1	2	3	4	5	6
1		1	1	1	1	1
2			2	3	4	5
3				3	4	5
4					4	5
5						5