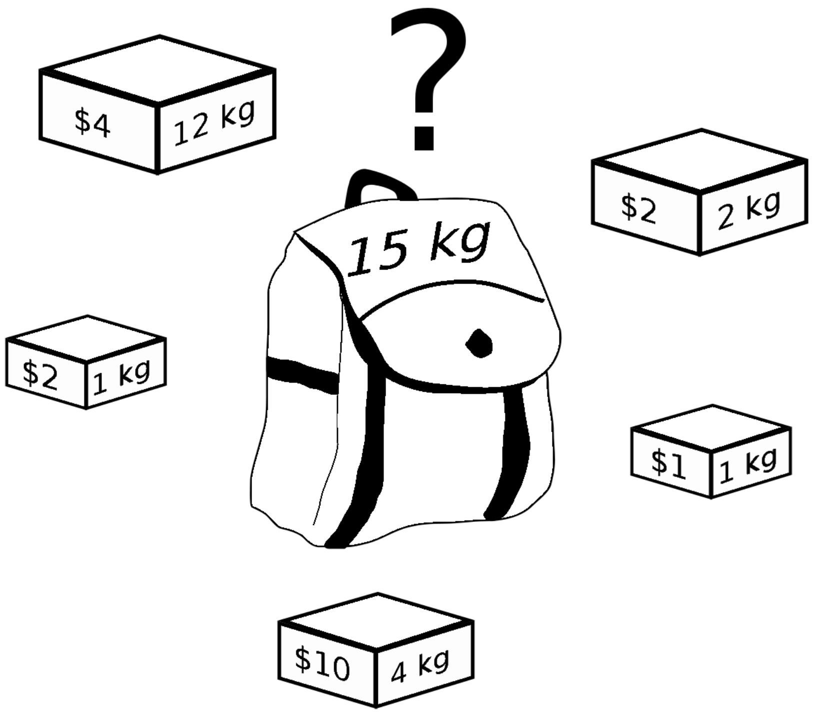


Dynamic Programming II

วิธีการแก้ปัญหาด้วย Dynamic Programming

ในการแก้ปัญหา DP จะมีขั้นตอนที่สามารถนำมาใช้แก้ได้ คือ

1.กำหนดปัญหาให้ชัดเจน (กำหนดเป็นฟังก์ชันว่าต้องรับข้อมูลอะไร และ return ข้อมูลอะไร ให้เอื้อกับการทำ recursive function)

2.หาคำตอบของปัญหาใหญ่จากปัญหาย่อย

3.เขียนสูตรออกมาในรูปทั่วไป (เขียน recurrence formula และกำหนด base case)

4.วิเคราะห์ Time Complexity และ Space Complexity

5.Implement! ทำได้สองวิธีคือ Top-down Approach และ Bottom-up Approach

ตัวอย่างที่1 : ปัญหา: 0/1 knapsack

- **ของ n ชิ้นมีหมายเลข :** $1, 2, 3, \ldots, n$
- **แต่ละชิ้นหนัก :** $w_1, w_2, w_3, \ldots, w_n$
- **แต่ละชิ้นมีมูลค่า :** $v_1, v_2, v_3, \ldots, v_n$
- **ถุงเป้หนึ่งใบจุของได้หนักไม่เกิน** W
- **ปัญหา : จงเลือกของใส่ถุง เพื่อให้**
  - ถุงไม่ขาด
  - ได้มูลค่ารวมมากสุด

นิยามปัญหา

- **ของ n ชิ้นมีหมายเลข :** $1, 2, 3, \ldots, n$
- **แต่ละชิ้นหนัก :** $w_1, w_2, w_3, \ldots, w_n$
- **แต่ละชิ้นมีมูลค่า :** $v_1, v_2, v_3, \ldots, v_n$
- **ถุงเป้หนึ่งใบจุของได้หนักไม่เกิน** W
- **หา** $<x_1, x_2, x_3, \ldots, x_n>$, $x_k = 0$ **หรือ** 1

$$\text{maximize } \sum_{k=1}^{n} x_k v_k$$

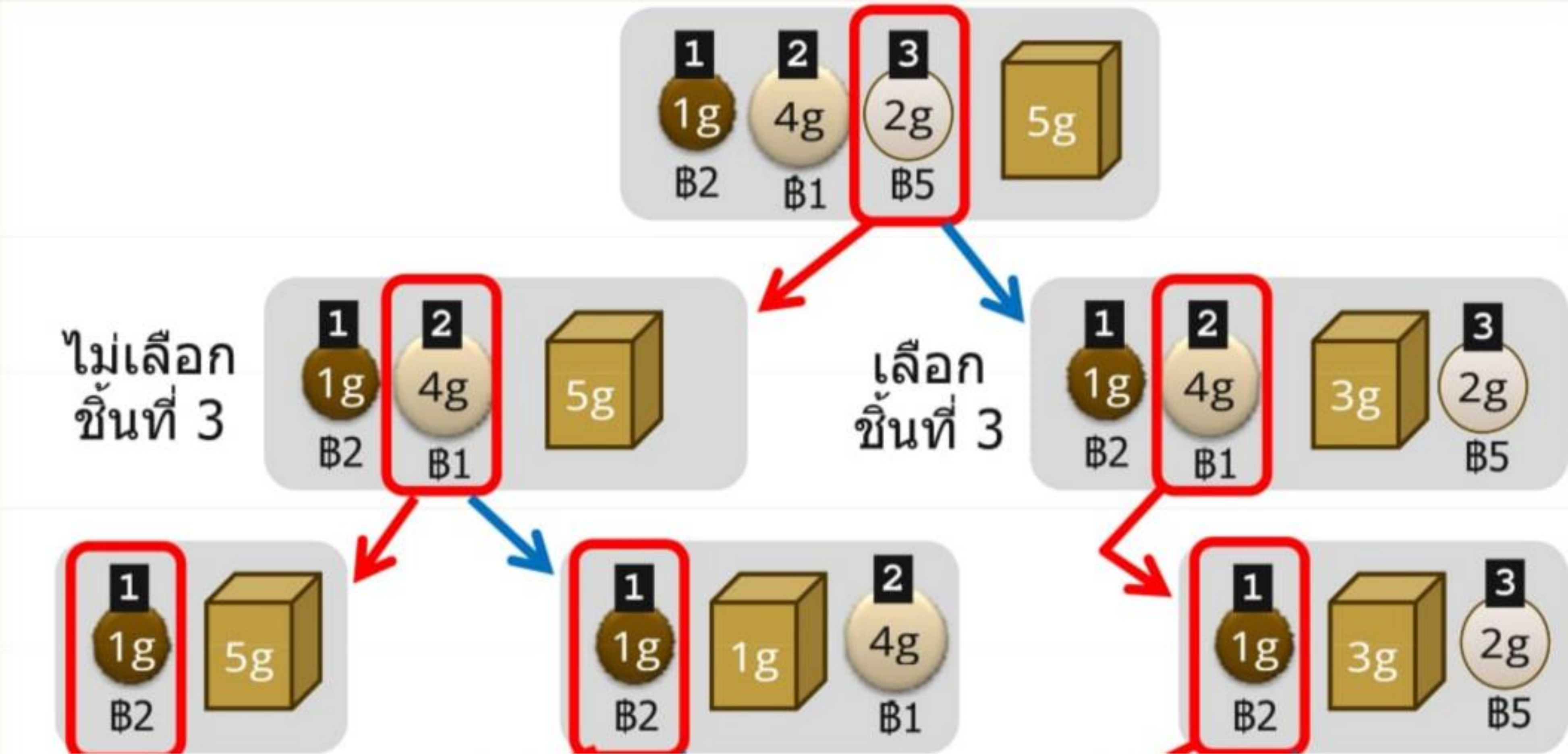
$$\text{subject to } \sum_{k=1}^{n} x_k w_k \le W$$

$$x_k \in \{0,1\}$$

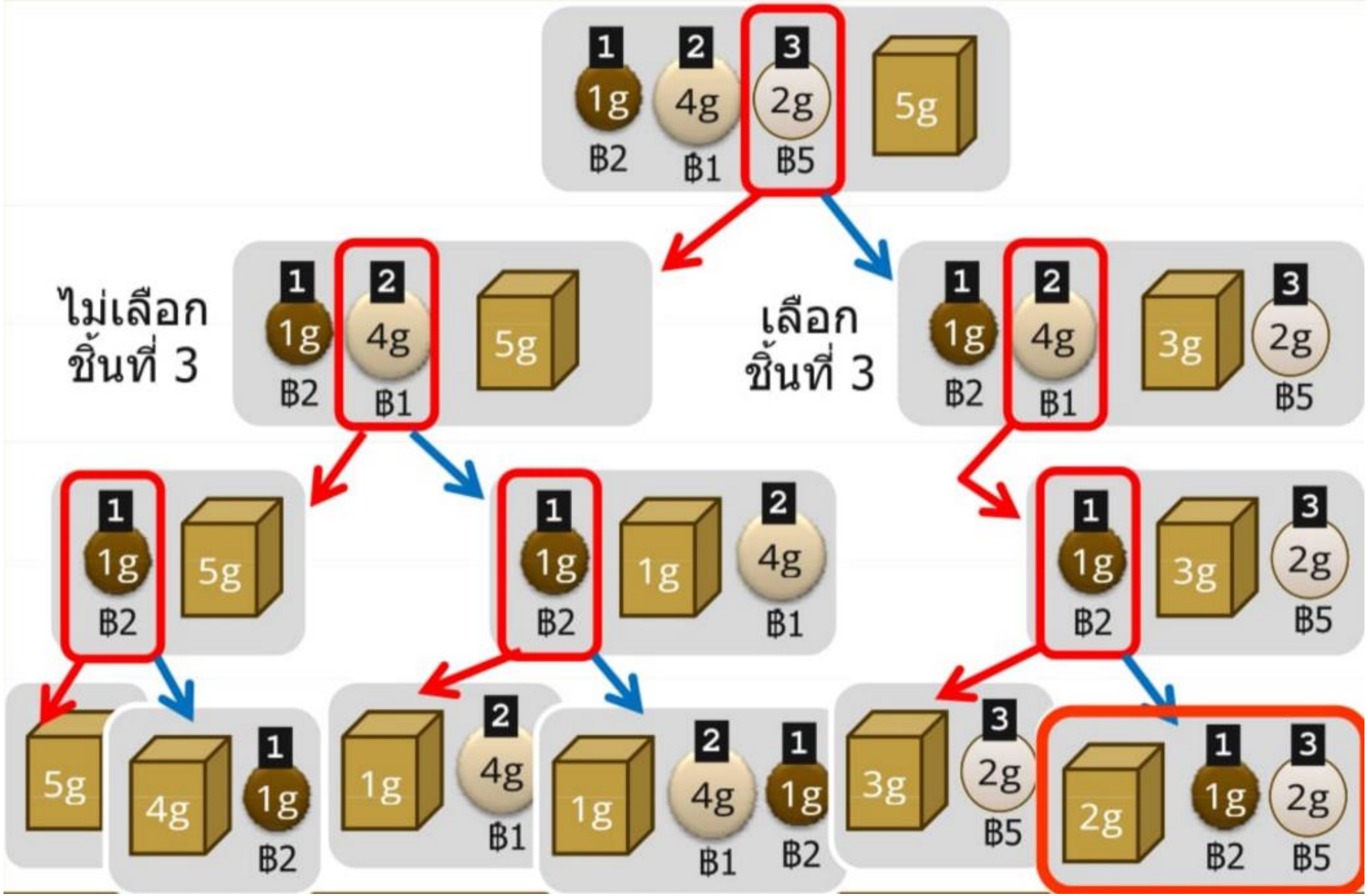
หาคำตอบของปัญหาใหญ่จากปัญหาย่อย

หาคำตอบของปัญหาใหญ่จากปัญหาย่อย

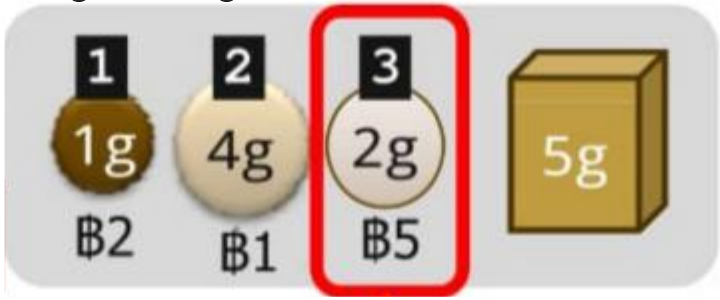
หาคำตอบของปัญหาใหญ่จากปัญหาย่อย

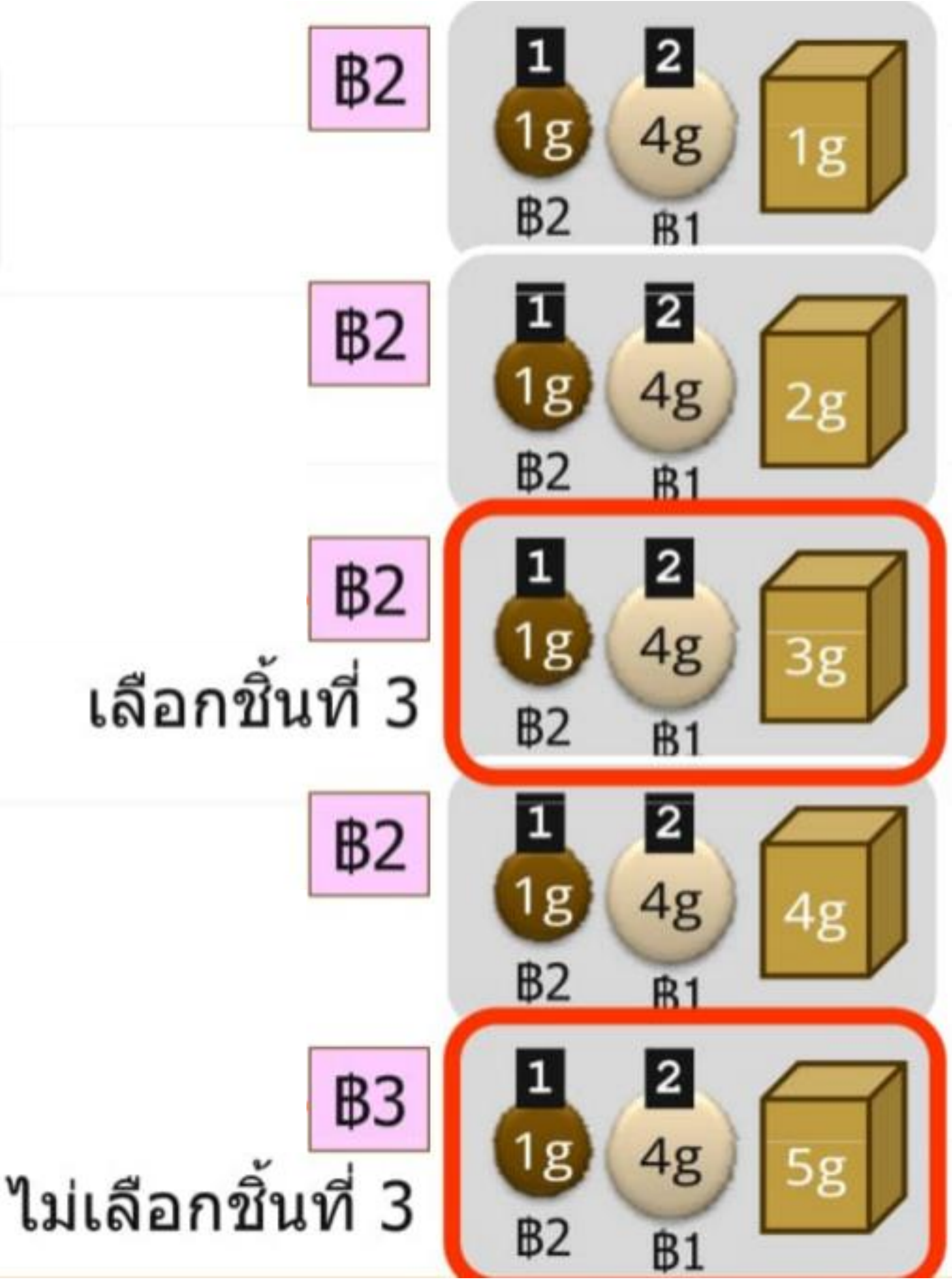


หาคำตอบของปัญหาใหญ่จากปัญหาย่อย

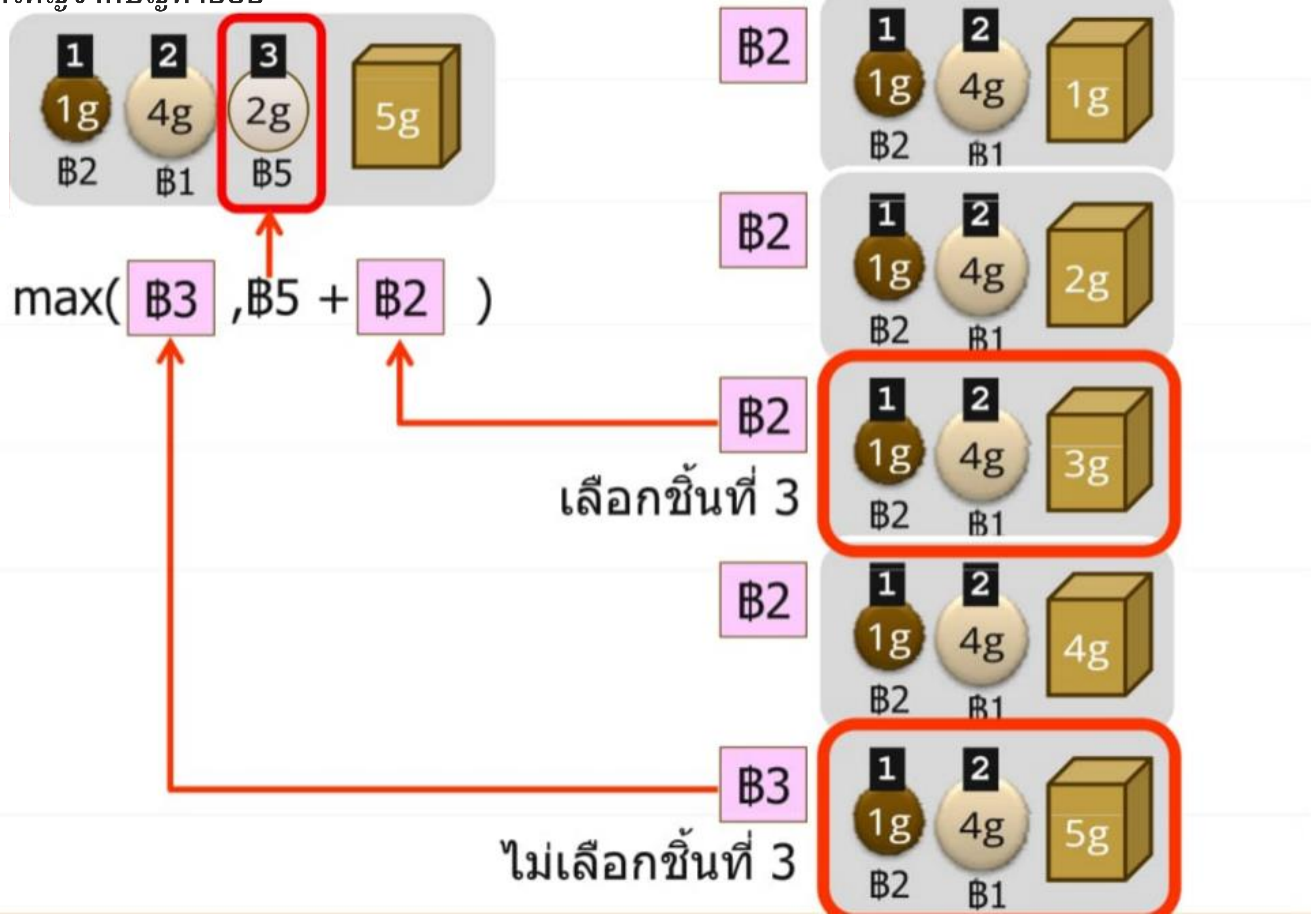


หาคำตอบของปัญหาใหญ่จากปัญหาย่อย

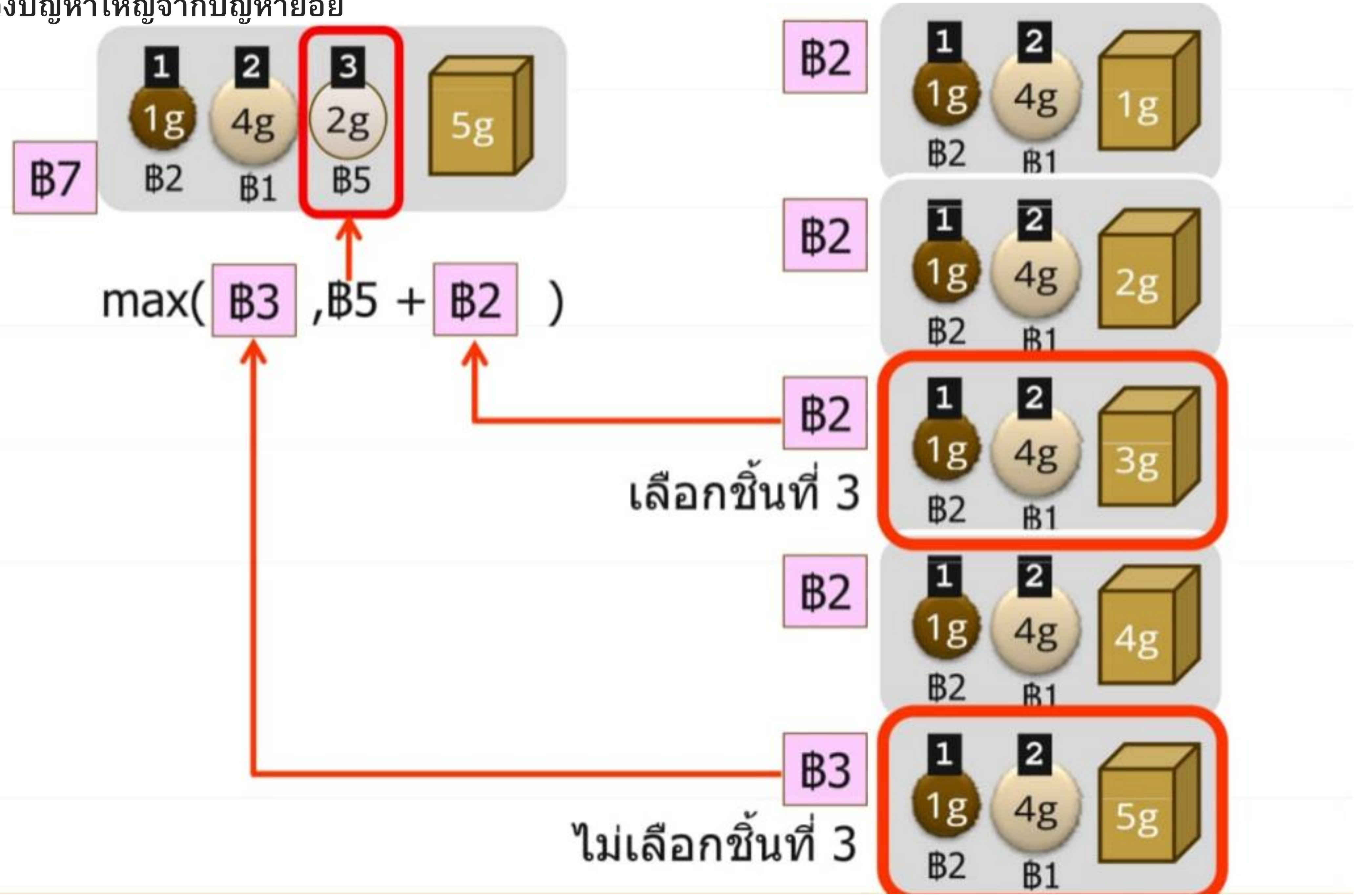




หาคำตอบของปัญหาใหญ่จากปัญหาย่อย



หาคำตอบของปัญหาใหญ่จากปัญหาย่อย



เขียนสูตรออกมาในรูปทั่วไป

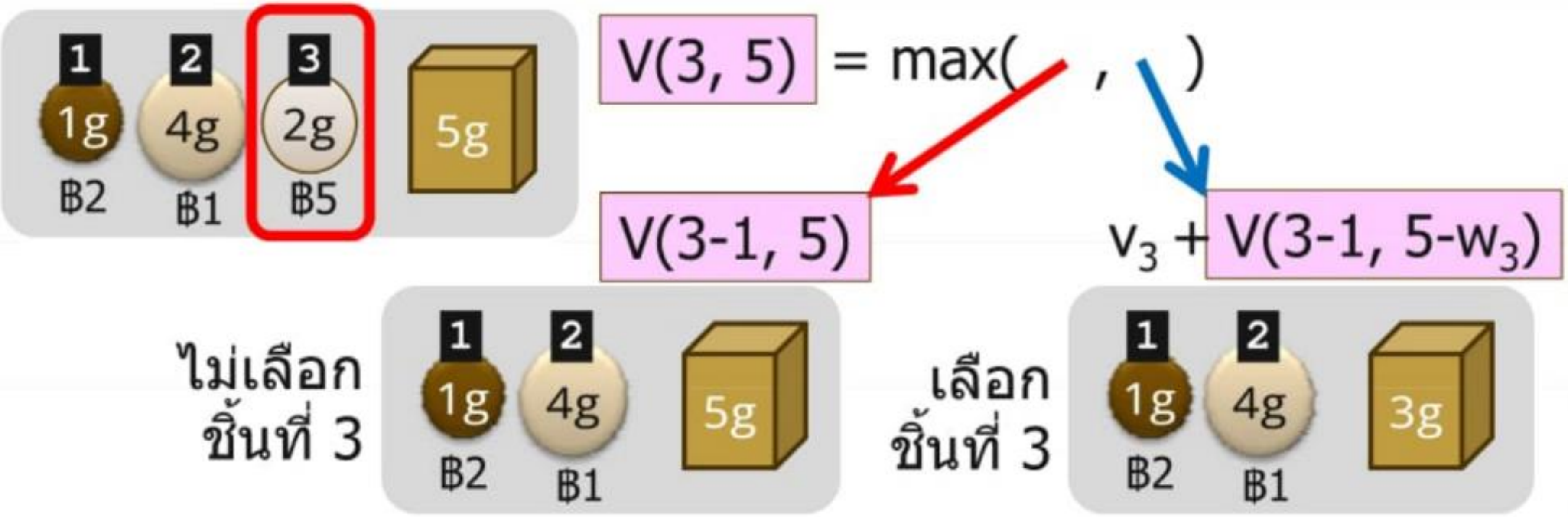
- **LCS** : $L(i, j)$
  - ปัญหาใหญ่หรือเล็กขึ้นกับความยาวของลำดับ X_i และ Y_j

- **Knapsack** : $V(\ ?\ )$
  - ปัญหาใหญ่หรือเล็กขึ้นกับจำนวนของ และน้ำหนักที่ถุงรับได้

$V(i, j)$ **: มูลค่ารวมสูงสุดในการ**
  - เลือกของชิ้นที่ $1, 2, 3, \ldots, i$
  - ใส่ถุงเป้ที่รับน้ำหนักได้ไม่เกิน j

เขียน recurrence ของ V(i, j)

เขียนสูตรออกมาในรูปทั่วไป



V(i, j) แทนมูลค่าของการเลือกที่ดีสุดเมื่อ

มีของให้เลือกชิ้นที่ 1, 2, ..., i ใส่ถุงเป้ที่จุได้หนัก j

$$V(i,j)=\begin{cases}\max\left(V(i-1,j),\ v_i+V(i-1,j-w_i)\right) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

Recurrence

$$V(i,j)=\begin{cases}\max\big(V(i-1,j),\, v_i+V(i-1,j-w_i)\big) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

```
V( v[1..n], w[1..n], W ) {
  return V(v, w, n, W)
}
V( v[1..n], w[1..n], i, j ) {
  if (i == 0 OR j == 0) return 0
  if (j < w[i]) {
    return V(v, w, i-1, j)
  } else {
    return max( V(v, w, i-1, j),
                V(v, w, i-1, j-w[i]) + v[i] )
  }
}
```

Recurrence + memo (Top-down)

$$V(i,j)=\begin{cases}\max\big(V(i-1,j),\, v_i+V(i-1,j-w_i)\big) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

```
V(v[1..n], w[1..n], i, j, M[1..n][1..W]) {
  if (i == 0 || j == 0) return 0
  if (M[i][j] > 0) return M[i][j]
  if (j < w[i]) {
    return M[i][j] = V(v, w, i-1, j)
  } else {
    return M[i][j] = max( V(v, w, i-1, j),
                          V(v, w, i-1, j-w[i]) )
  }
}
```

Bottom-up

$$V(i,j)=\begin{cases}\max\left(V(i-1,j),\ v_i+V(i-1,j-w_i)\right) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

v_i	w_i	i \ j	0	1	2	3	4	5	6	7	8	9	10
		0	0	0	0	0	0	0	0	0	0	0	0
20	2	1	0	0									
30	2	2	0	0									
66	3	3	0	0									
40	4	4	0	0									
60	5	5	0	0									

$W=10$

Bottom-up

$$V(i,j)=\begin{cases}\max\big(V(i-1,j),\ v_i+V(i-1,j-w_i)\big) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

v_i	w_i	i \ j	0	1	2	3	4	5	6	7	8	9	10
		0	0	0	0	0	0	0	0	0	0	0	0
20	2	1	0	0	20	20	20	20	20	20	20	20	20
30	2	2	0	0	30	30	50	50	50	50	50	50	50
66	3	3	0	0	30	66	66	96	96	116	116	116	116
40	4	4	0	0	30	66	66	96	96	116	116	136	136
60	5	5	0	0	30	66	66	96	96	116	126	136	156

$W = 10$

Bottom-up

```
knapsack_Value(v[1..n], w[1..n], W) {
  V = new array[0..n][0..W]
  for (i = 0; i <= n; i++) V[i][0] = 0
  for (j = 0; j <= W; j++) V[0][j] = 0
  for (i = 1; i <= n; i++)
    for (j = 1; j <= W; j++)
      if (j < w[i])
        V[i][j] = V[i-1][j]
      else
        V[i][j] = max(V[i-1][j]), v[i]+V[i-1][j-w[i]])
  return V
}
```

$\Theta(nW)$

$$V(i,j)=\begin{cases}\max\big(V(i-1,j),\, v_i+V(i-1,j-w_i)\big) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

Bottom-up: จำผลการตัดสินใจ

$$V(i,j)=\begin{cases}\max\left(V(i-1,j),\ v_i+V(i-1,j-w_i)\right) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

v_i	w_i	i \ j	0	1	2	3	4	5
		0	0	0	0	0	0	0
20	2	1	0	0 ✗	20 ✓	20 ✓	20 ✓	20 ✓
30	2	2	0	0 ✗	30 ✓	30 ✓	50 ✓	50 ✓
25	3	3	0	0 ✗	30 ✗	30 ✗	50 ✗	55 ✓

$W=5$

Bottom-up: จำผลการตัดสินใจ

v_i	w_i	i \ j	0	1	2	3	4	5
		0	0	0	0	0	0	0
20	2	1	0	✗	✓	✓	✓	✓
30	2	2	0	✗	✓	✓	✓	✓
25	3	3	0	✗	✗	✗	✗	✓

Bottom-up: จำผลการตัดสินใจ

```
knapsack(v[1..n], w[1..n], W) {
  V = new array[0..n][0..W]
  X = new array[1..n][1..W]
  for (i = 0; i <= n; i++) V[i][0] = 0
  for (j = 0; j <= W; j++) V[0][j] = 0
  for (i = 1; i <= n; i++)
    for (j = 1; j <= W; j++)
      if (j < w[i])
        V[i][j] = V[i-1][j]; X[i][j] = ☒
      else
        if ( V[i-1][j]) > v[i]+V[i-1][j - w[i]] ) {
          V[i][j] = V[i-1][j])
          X[i][j] = ☒
        } else {
          V[i][j] = v[i] + V[i-1][j - w[j]]
          X[i][j] = ☑
        }
```

Bottom-up: จำผลการตัดสินใจ

```
    S = an empty set
    i = n; j = W
    while (i > 0 AND j > 0) {
      if ( X[i][j] == ☑ ) {
        S.add(i);
        j = j - w[i];
      }
      i--
    }
    return S;
}
```

v_i	w_i	i \ j	0	1	2	3	4	5
		0	0	0	0	0	0	0
20	2	1	0	☒	☑	☑	☑	☑
30	2	2	0	☒	☑	☑	☑	☑
25	3	3	0	☒	☒	☒	☒	☑

$$V(i,j)=\begin{cases}\max\big(V(i-1,j),\ v_i+V(i-1,j-w_i)\big) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

v_i	w_i	i \ j	0	1	2	3	4	5	6	7	8	9	10
		0	0	0	0	0	0	0	0	0	0	0	0
20	2	1	0	0	20	20	20	20	20	20	20	20	20
30	2	2	0	0	30	30	50	50	50	50	50	50	50
66	3	3	0	0	30	66	66	96	96	116	116	116	116
40	4	4	0	0	30	66	66	96	96	116	116	136	136
60	5	5	0	0	30	66	66	96	96	116	126	136	156

$i=0$ or $j=0$? → false

$j<w_i$? → false

$V(i,j) = v_i + V(i-1, j-w_i)$? → true → เลือกชิ้นที่ 5

$$V(i,j)=\begin{cases}\max\big(V(i-1,j),\ v_i+V(i-1,j-w_i)\big) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

v_i	w_i	i \ j	0	1	2	3	4	5	6	7	8	9	10
		0	0	0	0	0	0	0	0	0	0	0	0
20	2	1	0	0	20	20	20	20	20	20	20	20	20
30	2	2	0	0	30	30	50	50	50	50	50	50	50
66	3	3	0	0	30	66	66	96	96	116	116	116	116
40	4	4	0	0	30	66	66	96	96	116	116	136	136
60	5	5	0	0	30	66	66	96	96	116	126	136	156

$i=0$ or $j=0$? → false

$j<w_i$? → false

$V(i,j) = v_i + V(i-1, j-w_i)$? → false

$$
V(i,j)=\begin{cases}\max\big(V(i-1,j),\ v_i+V(i-1,j-w_i)\big) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}
$$

v_i	w_i	i \ j	0	1	2	3	4	5	6	7	8	9	10
		0	0	0	0	0	0	0	0	0	0	0	0
20	2	1	0	0	20	20	20	20	20	20	20	20	20
30	2	2	0	0	30	30	50	50	50	50	50	50	50
66	3	3	0	0	30	66	66	96	96	116	116	116	116
40	4	4	0	0	30	66	66	96	96	116	116	136	136
60	5	5	0	0	30	66	66	96	96	116	126	136	156

$i = 0$ or $j = 0$? → false

$j < w_i$? → false

$V(i, j) = v_i + V(i-1, j-w_i)$? → true → เลือกชิ้นที่ 3

$$V(i,j)=\begin{cases}\max\big(V(i-1,j),\ v_i+V(i-1,j-w_i)\big) & \text{if } i>0 \text{ and } j\ge w_i\\ V(i-1,j) & \text{if } j<w_i\\ 0 & \text{if } i=0 \text{ or } j=0\end{cases}$$

v_i	w_i	i \ j	0	1	2	3	4	5	6	7	8	9	10
		0	0	0	0	0	0	0	0	0	0	0	0
20	2	1	0	0	20	20	20	20	20	20	20	20	20
30	2	2	0	0	30	30	50	50	50	50	50	50	50
66	3	3	0	0	30	66	66	96	96	116	116	116	116
40	4	4	0	0	30	66	66	96	96	116	116	136	136
60	5	5	0	0	30	66	66	96	96	116	126	136	156

$i=0$ or $j=0$? → false

$j<w_i$? → false

$V(i,j) = v_i + V(i-1, j-w_i)$? → true → เลือกชิ้นที่ 2

$$V(i,j) = \begin{cases} \max\big(V(i-1,j),\ v_i + V(i-1, j-w_i)\big) & \text{if } i>0 \text{ and } j \ge w_i \\ V(i-1,j) & \text{if } j < w_i \\ 0 & \text{if } i=0 \text{ or } j=0 \end{cases}$$

v_i	w_i	i \ j	0	1	2	3	4	5	6	7	8	9	10
		0	0	0	0	0	0	0	0	0	0	0	0
20	2	1	0	0	20	20	20	20	20	20	20	20	20
30	2	2	0	0	30	30	50	50	50	50	50	50	50
66	3	3	0	0	30	66	66	96	96	116	116	116	116
40	4	4	0	0	30	66	66	96	96	116	116	136	136
60	5	5	0	0	30	66	66	96	96	116	126	136	156

$i = 0$ or $j = 0$? $\rightarrow$ true

Bottom-up: จำผลการตัดสินใจ

```
knapsack_Soln( v[1..n], w[1..n], V[0..n][0..W] ) {
  S = an empty set
  i = n; j = W
  while (i > 0 AND j > 0) {
    if (j >= w[i] AND
          V[i][j] == v[i] + V[i-1][j - w[i]]) {
      S.add(i);
      j = j - w[i];
    }
    i--
  }
  return S;
}
```

ตัวอย่างที่ 2 : Quick-sum, Prefix-sum, Cumulative Sum

1D:

Original Array

1	2	3	4	5	6

Cumulative Sum Array

1	3	6	10	15	21

```c
int main() {
    scanf("%d", &n);
    for(int i = 1; i <= n; ++i) scanf("%d", &dp[i]), dp[i] += dp[i-1];
    scanf("%d", &m);
    for(int i = 1;i <= m; ++i) {
        scanf("%d %d", &l, &r);
        printf("%d\n", dp[r] - dp[l-1]);
    }
}
```

2D:

$$\begin{bmatrix} 10 & 20 & 30 \\ 5 & 10 & 20 \\ 2 & 4 & 6 \end{bmatrix} \qquad \begin{bmatrix} 10 & 30 & 60 \\ 15 & 45 & 95 \\ 17 & 51 & 107 \end{bmatrix}$$

Input **Prefix Sum**

```c
int main() {
    scanf("%d", &n);
    for(int i = 1; i <= n; ++i) {
        for(int j = 1; j <= n; ++j) {
            scanf("%d", &dp[i][j]);
            dp[i][j] += dp[i-1][j] + dp[i][j-1] - dp[i-1][j-1];
        }
    }
    scanf("%d", &m);
    for(int i = 1; i <= m; ++i) {
        scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
        printf("%d\n", dp[x2][y2] - dp[x1-1][y1] - dp[x2][y1-1] + dp[x1-1][y1-1]);
    }
}
```

ตัวอย่างที่ 3 Matrix chain multiplication (MCM)

$$A_1A_2A_3A_4$$

$$(A_1(A_2(A_3A_4)))$$
$$(A_1((A_2A_3)A_4))$$
$$((A_1A_2)(A_3A_4))$$
$$(((A_1A_2)A_3)A_4)$$
$$((A_1(A_2A_3))A_4)$$

วงเล็บกำหนดลำดับการคูณ

ลำดับการคูณต่างกันใช้
เวลาคูณต่างกัน

จงหาวิธีการใส่วงเล็บ
ที่ใช้เวลาการคูณเร็วสุด

Matrix chain multiplication (MCM)

$$\underset{[p \times r]}{C} = \underset{[p \times q]}{A} \times \underset{[q \times r]}{B}$$

ต้องคูณด้วย *
จำนวน pqr ครั้ง

```
mult(A[1..p][1..q], B[1..q][1..r]) {
  create C[1..p][1..r]
  for (i = 1; i <= p; i++) {
    for (j = 1; j <= r; j++) {
      C[i][j] = 0
      for (k = 1; k <= q; k++) {
        C[i][j] += A[i][k] * B[k][j];
      }
    }
  }
  return C;
}
```

$$c_{i,j} = \sum_{k=1}^{q} a_{i,k} b_{k,j}$$

- **A_1 A_2 A_3**
  [10 x 100], [100 x 5], [5 x 50]
- **คูณตามลำดับ $((A_1A_2)A_3)$**
  - (A_1A_2) ต้องคูณด้วย * จำนวน 10 x 100 x 5 = 5,000 ครั้ง
  - ได้เมทริกซ์ A_{12} ขนาด [10 x 5]
  - $(A_{12}A_3)$ ต้องคูณด้วย * จำนวน 10 x 5 x 50 = 2,500 ครั้ง
  - รวมเป็น = 7,500 ครั้ง
- **คูณตามลำดับ $(A_1(A_2A_3))$**
  - (A_2A_3) ต้องคูณด้วย * จำนวน 100 x 5 x 50 = 25,000 ครั้ง
  - ได้เมทริกซ์ A_{23} ขนาด [100 x 50]
  - (A_1A_{23}) ต้องคูณด้วย * จำนวน 10 x 100 x 50 = 50,000 ครั้ง
  - รวมเป็น = 75,000 ครั้ง

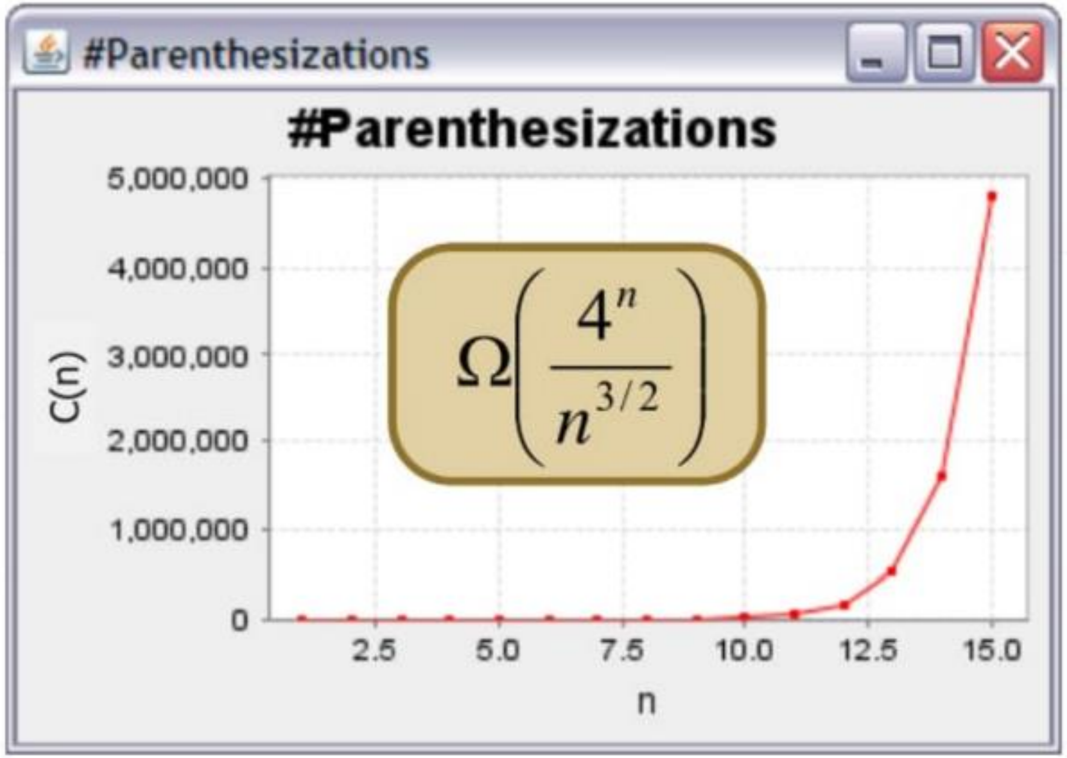
$$
\begin{array}{lllll}
A_1 & A_2 & A_3 & A_4 & A_5 \\
(A_1) & (A_2 & A_3 & A_4 & A_5) \quad \rightarrow 1 \times 5 \\
(A_1 & A_2) & (A_3 & A_4 & A_5) \quad \rightarrow 1 \times 2 \\
(A_1 & A_2 & A_3) & (A_4 & A_5) \quad \rightarrow 2 \times 1 \\
(A_1 & A_2 & A_3 & A_4) & (A_5) \quad \rightarrow 5 \times 1
\end{array}
$$

$C(n)$ คือจำนวนการใส่วงเล็บกรณีมี n ตัว

$$
\underbrace{(A_1 \ \ldots \ A_k)}_{C(k)} \ \underbrace{(A_{k+1} \ \ldots \ A_n)}_{C(n-k)}
$$

$$
\boxed{C(n) = \sum_{k=1}^{n-1} \left(C(k)\, C(n-k) \right)}
$$

$$C(n) = \begin{cases} \sum_{k=1}^{n-1} \left(C(k)C(n-k) \right) & n \geq 3 \\ 1 & n \leq 2 \end{cases}$$



ช้ามาก ๆ ถ้าต้องลองทุกรูปแบบ

Matrix chain multiplication (MCM): นิยามปัญหา

- ต้องการหาวิธีใส่วงเล็บของ $A_1\ A_2 \ldots A_n$
- A_1 มีขนาด $p_0 \times p_1$
- A_2 มีขนาด $p_1 \times p_2$

  ...
- A_n มีขนาด $p_{n-1} \times p_n$
- $A_i \ldots A_j$ มีขนาด $p_{i-1} \times p_j$
- ยังไม่หาวิธีการใส่วงเล็บที่ดีสุด
- ขอหาจำนวนการคูณด้วย * ของการใส่วงเล็บที่ดีสุด
- $m(i, j)$ =จำนวน * น้อยสุดเพื่อหาผลคูณ $A_i \ldots A_j$
- $m(1, n)$ คือคำตอบที่ต้องการ

Matrix chain multiplication (MCM): หาคำตอบของปัญหาใหญ่จากปัญหาย่อย

$m(i, j)$ คือจำนวนการคูณด้วย * น้อยสุดของการคูณ $\mathbf{A_i}\ \mathbf{A_{i+1}} \ldots \mathbf{A_j}$

คูณ $\mathbf{A_i} \quad \ldots \quad \mathbf{A_k}$ ใช้ $m(i, k)$ ได้เมทริกซ์ขนาด $[\, p_{i-1} \times p_k \,]$

คูณ $\mathbf{A_{k+1}} \quad \ldots \quad \mathbf{A_j}$ ใช้ $m(k+1, j)$ ได้เมทริกซ์ขนาด $[\, p_k \times p_j \,]$

คูณเมทริกซ์ทั้งสองใช้ $p_{i-1} p_k p_j$

$$\underbrace{\overset{[\, p_{i-1} \times p_k \,]}{(\mathbf{A_i} \quad \ldots \quad \mathbf{A_k})}}\ \underbrace{\overset{[\, p_k \times p_j \,]}{(\mathbf{A_{k+1}} \quad \ldots \quad \mathbf{A_j})}}$$

$$\boxed{m(i, k) \quad + \quad m(k+1, j) \quad + \quad p_{i-1} p_k p_i}$$

จำนวนการคูณด้วย * น้อยสุด เมื่อแบ่งที่ k

Matrix chain multiplication (MCM): เขียนสูตรออกมาในรูปทั่วไป

$m(i, j)$ คือจำนวนการคูณด้วย * น้อยสุดของการคูณ $\mathbf{A}_i\ \mathbf{A}_{i+1} \ldots \mathbf{A}_j$

$$\underbrace{(\mathbf{A}_i \;\ldots\; \mathbf{A}_k)}_{m(i,k)} + \underbrace{(\mathbf{A}_{k+1} \;\ldots\; \mathbf{A}_j)}_{m(k+1,j)} + p_{i-1}\, p_k\, p_j$$

$(\mathbf{A}_1)\ (\mathbf{A}_2\ \mathbf{A}_3\ \mathbf{A}_4\ \mathbf{A}_5)$ $\quad m(1,1) + m(2,5) + p_0 p_1 p_5$

$(\mathbf{A}_1\ \mathbf{A}_2)\ (\mathbf{A}_3\ \mathbf{A}_4\ \mathbf{A}_5)$ $\quad m(1,2) + m(3,5) + p_0 p_2 p_5$

$(\mathbf{A}_1\ \mathbf{A}_2\ \mathbf{A}_3)\ (\mathbf{A}_4\ \mathbf{A}_5)$ $\quad m(1,3) + m(4,5) + p_0 p_3 p_5$

$(\mathbf{A}_1\ \mathbf{A}_2\ \mathbf{A}_3\ \mathbf{A}_4)\ (\mathbf{A}_5)$ $\quad m(1,4) + m(5,5) + p_0 p_4 p_5$

$$m(i,j) = \begin{cases} \min\limits_{i \le k \le j-1} \left\{ m(i,k) + m(k+1,j) + p_{i-1} p_k p_j \right\} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

Matrix chain multiplication (MCM): Recursive

$$m(i,j) = \begin{cases} \min\limits_{i \le k \le j-1} \left\{ m(i,k) + m(k+1,j) + p_{i-1} p_k p_j \right\} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

```
mcm( p[0..n], i, j ) {
  if (i == j) return 0;

  minMCM = ∞;
  for( k = i; k <= j-1; k++ ) {
    minMCM = min( minMCM,
                  mcm(p, i, k) +
                  mcm(p, k+1, j) +
                  p[i-1]*p[k]*p[j] );
  }
  return minMCM;
}
```

Matrix chain multiplication (MCM): Top-down (Recursive+Memo)

$$m(i,j) = \begin{cases} \min\limits_{i \le k \le j-1} \left\{ m(i,k) + m(k+1,j) + p_{i-1} p_k p_j \right\} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

```
mcm( p[0..n], i, j , M[1..n][1..n] ) {
  if (i == j) return 0;
  if (M[i][j] > 0) return M[i][j]
  M[i][j] = ∞
  for( k = i; k <= j-1; k++ ) {
    M[i][j]= min( M[i][j],
                  mcm(p, i, k, M) +
                  mcm(p, k+1, j, M) +
                  p[i-1]*p[k]*p[j] );
  }
  return M[i][j]
}
```

Matrix chain multiplication (MCM): Bottom-up

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$

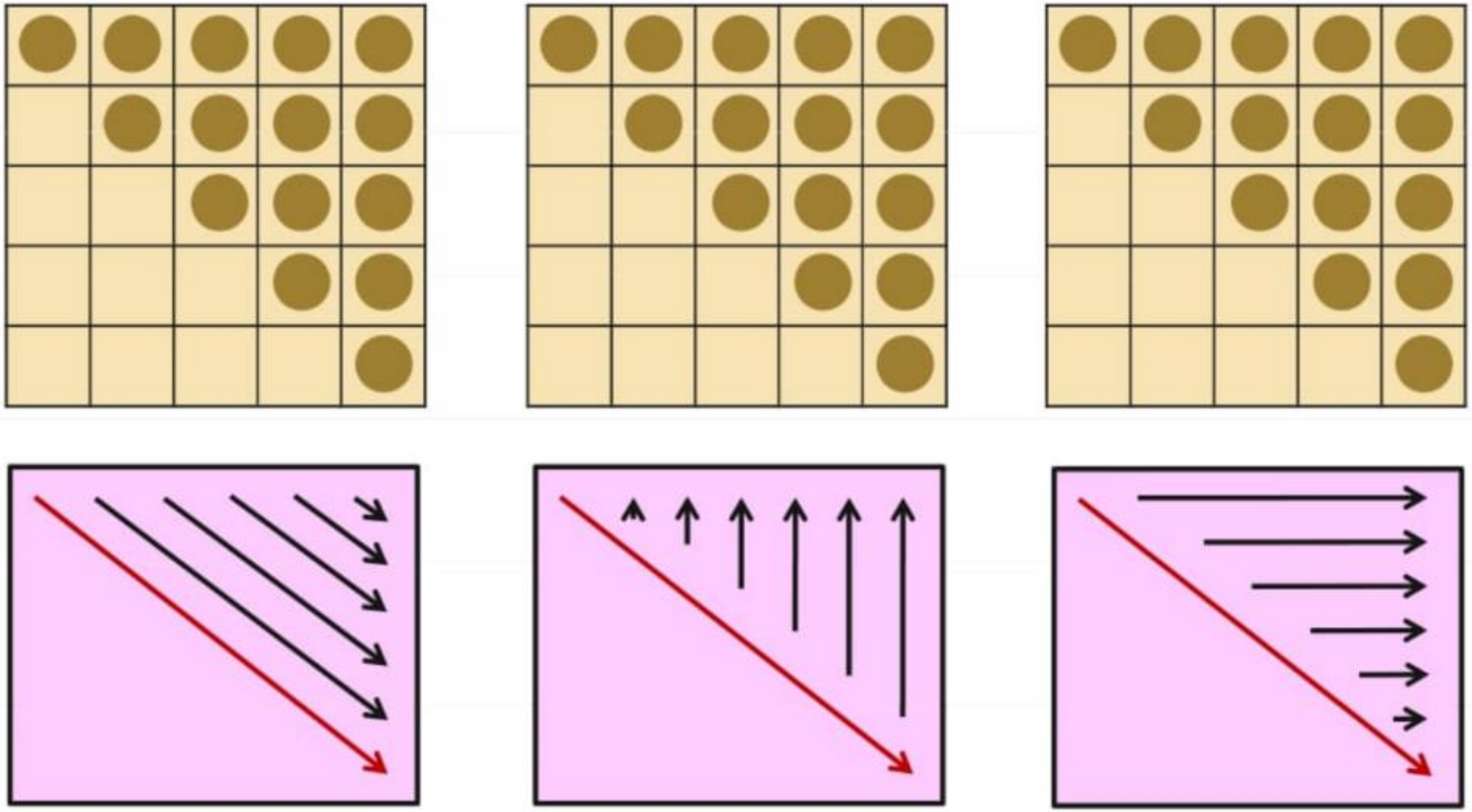
i \ j	1	2	3	4	5
1					
2		m(2,2)	m(2,3)	m(2,4)	m(2,5)
3					m(3,5)
4					m(4,5)
5					m(5,5)

Matrix chain multiplication (MCM): Bottom-up

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$

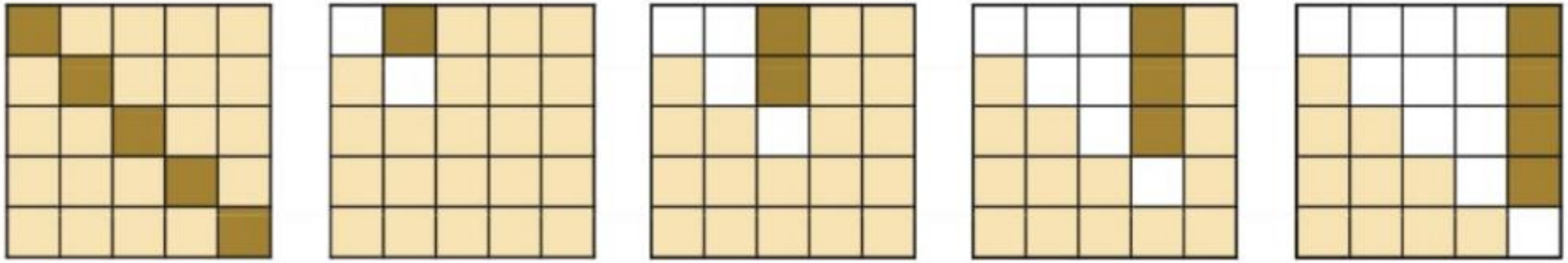
i \ j	1	2	3	4	5
1	m(1,1)	m(2,2)	m(2,3)	m(2,4)	m(1,5)
2					m(2,5)
3					m(3,5)
4					m(4,5)
5					m(5,5)

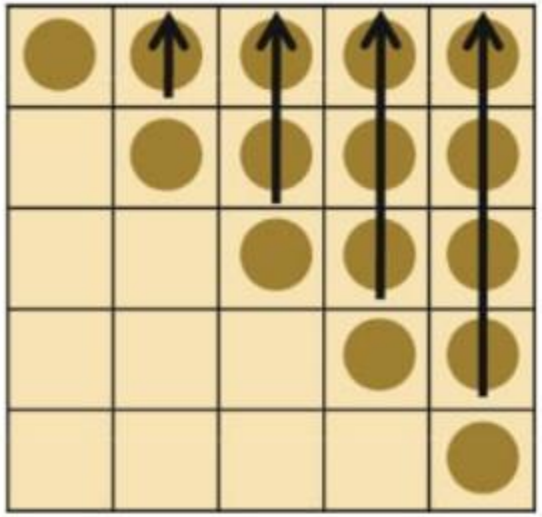
เติมคำตอบเล็กก่อนคำตอบใหญ่



Matrix chain multiplication (MCM): Bottom-up - รูปแบบการเติมตาราง

$$m(i,j) = \begin{cases} \min\limits_{i \le k \le j-1} \left\{ m(i,k) + m(k+1,j) + p_{i-1} p_k p_j \right\} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

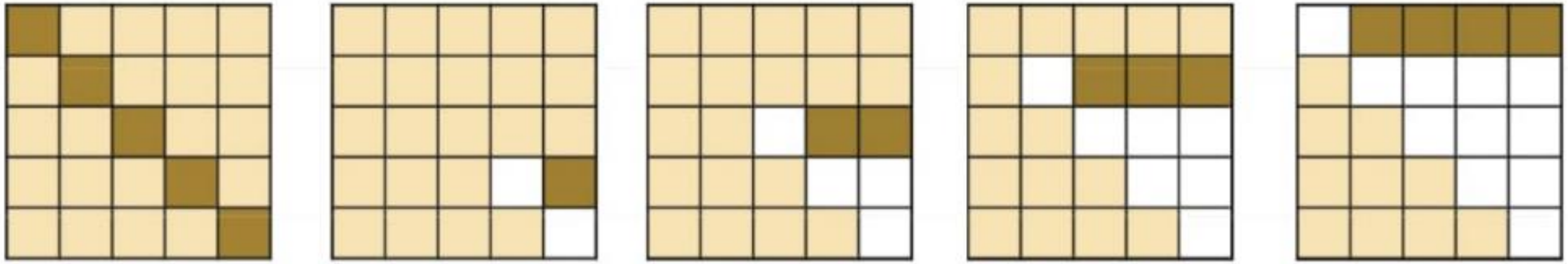


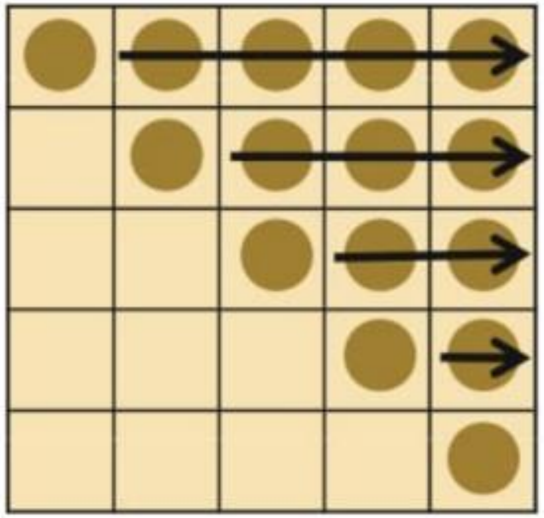


```
for( i = 1; i <= n; i++ )
  m[i][i] = 0
for( j = 2; j <= n; j++ ) {
  for ( i = j-1; i >= 1; i-- ) {
    m[i][j] = ...
  }
}
```

Matrix chain multiplication (MCM): Bottom-up - รูปแบบการเติมตาราง

$$m(i,j) = \begin{cases} \min\limits_{i \le k \le j-1} \left\{ m(i,k) + m(k+1,j) + p_{i-1} p_k p_j \right\} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

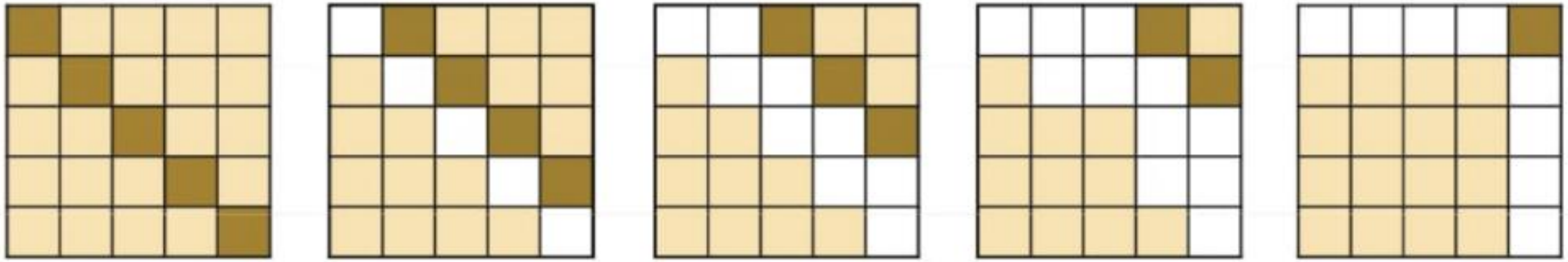


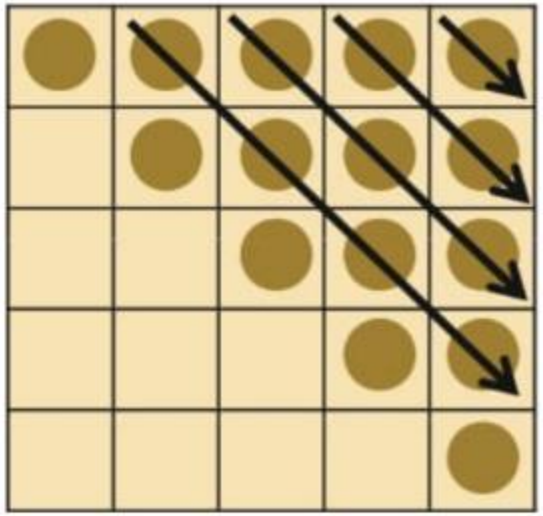


```
for( i = 1; i <= n; i++ )
  m[i][i] = 0
for( i = n-1; i >= 1; i-- ) {
  for ( j = i+1; j <= n; j++ ) {
    m[i][j] = ...
  }
}
```

Matrix chain multiplication (MCM): Bottom-up - รูปแบบการเติมตาราง

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{ m(i,k)+m(k+1,j)+p_{i-1}p_kp_j \right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$





ตัวอย่าง

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$

$$\begin{array}{cccccc} & A_1 & A_2 & A_3 & A_4 & A_5\end{array}$$

$$10 \times 5 \times 1 \times 5 \times 10 \times 2$$

i \ j	1	2	3	4	5
1	0	50			
2		0	25		
3			0	50	
4				0	100
5					0

ตัวอย่าง

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{ m(i,k)+m(k+1,j)+p_{i-1}p_kp_j \right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$

	A_1	A_2	A_3	A_4	A_5

10 × 5 × 1 × 5 × 10 × 2

i \ j	1	2	3	4	5
1	0	50	100		
2		0	25		
3			0	50	
4				0	100
5					0

0 + 25 + 10x5x5 = 275

50 + 0 + 10x1x5 = $\boxed{100}$

ตัวอย่าง

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$

$$\begin{array}{cccccc} & A_1 & A_2 & A_3 & A_4 & A_5\\ 10 \times & 5 \times & 1 \times & 5 \times & 10 \times & 2\end{array}$$

i \ j	1	2	3	4	5
1	0	50	100		
2		0	25	100	
3			0	50	
4				0	100
5					0

0 + 50 + 5x1x10 = $\boxed{100}$

25 + 0 + 5x5x10 = 275

ตัวอย่าง

$$m(i,j) = \begin{cases} \min\limits_{i \le k \le j-1} \{ m(i,k) + m(k+1,j) + p_{i-1} p_k p_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

	A_1	A_2	A_3	A_4	A_5
10 ×	5 ×	1 ×	5 ×	10 ×	2

i \ j	1	2	3	4	5
1	0	50	100		
2		0	25	100	
3			0	50	70
4				0	100
5					0

0 + 100 + 1x5x2 = 110

50 + 0 + 1x10x2 = $\boxed{70}$

ตัวอย่าง

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{ m(i,k)+m(k+1,j)+p_{i-1}p_k p_j \right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$

A_1	A_2	A_3	A_4	A_5
10 × 5	5 × 1	1 × 5	5 × 10	10 × 2

10 × 5 × 1 × 5 × 10 × 2

i \ j	1	2	3	4	5
1	0	50	100	$\boxed{200}$	
2		0	25	100	
3			0	50	70
4				0	100
5					0

0 + 100 + 10x5x10 = 600

50 + 50 + 10x1x10 = $\boxed{200}$

100 + 0 + 10x5x10 = 600

ตัวอย่าง

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$

$$\begin{array}{cccccc} & A_1 & A_2 & A_3 & A_4 & A_5 \\ 10\,\times & 5\,\times & 1\,\times & 5\,\times & 10\,\times & 2\end{array}$$

i \ j	1	2	3	4	5
1	0	50	100	200	
2		0	25	100	80
3			0	50	70
4				0	100
5					0

0 + 70 + 5x1x2 = $\boxed{80}$

25 + 100 + 5x5x2 = 175

100 + 0 + 5x10x2 = 200

ตัวอย่าง

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{ m(i,k)+m(k+1,j)+p_{i-1}p_k p_j \right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$

$$\begin{array}{cccccc} & A_1 & A_2 & A_3 & A_4 & A_5 \\ 10\times & 5\times & 1 & \times 5 & \times 10 & \times 2\end{array}$$

i \ j	1	2	3	4	5
1	0	50	100	200	140
2		0	25	100	80
3			0	50	70
4				0	100
5					0

0 + 80 + 10x5x2 = 180

50 + 70 + 10x1x2 = $\boxed{140}$

100 + 100 + 10x5x2 = 300

200 + 0 + 10x10x2 = 400

ลูปของการเติมตาราง

$$m(i,j)=\begin{cases}\min\limits_{i\le k\le j-1}\left\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\right\} & \text{if } i<j\\ 0 & \text{if } i=j\end{cases}$$

```
for( i = 1; i <= n; i++) M[i][i]=0
for( d = 1; d < n; d++ ) {
  for ( i = 1; i <= n-d; i++ ) {
    j = i + d;
    for ( k = i; k < j; k++ ) {
      M[i][j] = ...
    }
  }
}
```

1,1	1,2	1,3	1,4	1,5
	2,2	2,3	2,4	2,5
		3,3	3,4	3,5
			4,4	4,5
				5,5

d = 1 i, j	d = 2 i, j	d = 3 i, j	d = 4 i, j
1, 2	1, 3	1, 4	1, 5
2, 3	2, 4	2, 5	
3, 4	3, 5		
4, 5			

Matrix chain multiplication (MCM): Bottom-up

$$m(i,j) = \begin{cases} \min\limits_{i \le k \le j-1} \left\{ m(i,k) + m(k+1,j) + p_{i-1} p_k p_j \right\} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

```
mcm_Value( p[0..n] ) {
  M = new array[1..n][1..n]
  for( i = 1; i <= n; i++ ) M[i][i] = 0
  for( d = 1; d < n; d++ ) {
    for ( i = 1; i <= n-d; i++ ) {
      j = i + d;
      M[i][j] = ∞
      for ( k = i; k < j; k++ ) {
        M[i][j] = min( M[i][j],
           M[i][k] + M[k+1][j] + p[i-1]*p[k]*p[j]);
      }
    }
  }
  return M;
}
```

$\Theta(n^3)$

Matrix chain multiplication (MCM): Bottom-up - จำจุดแบ่ง k ที่ได้ค่าน้อยสุด

i \ j	1	2	3	4	5
1	0	1 50	1 100	2 200	2 140
2		0	2 25	2 100	2 80
3			0	3 50	4 70
4				0	4 100
5					0

$$(\mathbf{A}_i \ \ldots \ \mathbf{A}_k)(\mathbf{A}_{k+1} \ \ldots \ \mathbf{A}_j)$$

$$m(i,j) = \begin{cases} \min\limits_{i \le k \le j-1} \{ m(i,k) + m(k+1,j) + p_{i-1} p_k p_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

Matrix chain multiplication (MCM): Bottom-up - การใส่วงเล็บที่คูณเร็วสุด

i \ j	1	2	3	4	5
1	0	1 50	1 100	2 200	2 140
2		0	2 25	2 100	2 80
3			0	3 50	4 70
4				0	4 100
5					0

[1,5] → 2 $(A_1 \; A_2)(A_3 \; A_4 \; A_5)$

[1,2] → 1 $((A_1)(A_2))(A_3 \; A_4 \; A_5)$

[3,5] → 4 $((A_1)(A_2))((A_3 \; A_4)(A_5))$

[3,4] → 3 $((A_1)(A_2))(((A_3)(A_4))(A_5))$

Matrix chain multiplication (MCM): จำจุดแบ่ง matrix chain

```
mcm_Order ( p[0..n] ) {
  M = new Array[1..n][1..n]
  K = new Array[1..n][1..n]
  for( d = 1; d < n; d++ ) {
    for ( i = 1; i <= n-d; i++ ) {
      j = i + d;
      M[i][j] = ∞
      for ( k = i; k < j; k++ ) {
        t = M[i][k] + M[k+1][j] + p[i-1]*p[k]*p[j];
        if (t < M[i][j]) {
          M[i][j] = t;
          K[i][j] = k;
        }
      }
    }
  }
  return K;
}
```

$(A_i \;...\; A_k)(A_{k+1} \;...\; A_j)$

Matrix chain multiplication (MCM):

```
mcm_Mult( A[1..n], p[0..n] ) {
  K = mcm_Order( p );
  return mcm_Mult( A, K, 1, n );
}
```

$\Theta(n^3)$

$\Theta(\,?\,)$

```
mcm Mult( A[1..n], K[1..n][1..n], i, j ) {
  if (i == j) return A[i];
  X = mcm_Mult( A, K, i, K[i][j] );
  Y = mcm_Mult( A, K, K[i][j]+1, j );
  return mult(X, Y);
}
```

$$(A_i \; ... \; A_k)\,(A_{k+1} \; ... \; A_j)$$

$$X \quad \times \quad Y$$

Reference

https://www.cp.eng.chula.ac.th/~somchai/books/index.html

https://www.geeksforgeeks.org/dynamic-programming/