

เอกสารประกอบการอบรม

เรื่อง Dynamic Programming Algorithms

ผู้ช่วยศาสตราจารย์ ดร.ณัฐพงศ์ ชินธเนศ

นำมาจาก Slide วิชา Algorithms

โดย รองศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล

Dynamic Programming

[Dynamic Programming Slides](#)

Easy

[Edit Distance](#) <http://www.spoj.com/problems/EDIST/>

[Matrix Chain](#) <http://www.spoj.com/problems/MIXTURES/>

[Sucuba](#) <http://www.spoj.com/problems/SCUBADIV/>

Problems for Discussion

[Star Adventure](#) <https://www.topcoder.com/community/data-science/data-science-tutorials/dynamic-programming-from-novice-to-advanced/>

[Mini Paint](#) https://community.topcoder.com/stat?c=problem_statement&pm=1996&rd=4710

[Theater](#)

Hard

[Mars Stomatology](#) <http://acm.sgu.ru/problem.php?contest=0&problem=304>

[Bit](#) <http://acm.sgu.ru/problem.php?contest=0&problem=269>

[Fast Ride](#) <http://acm.sgu.ru/problem.php?contest=0&problem=317>

[Mono chrome picture](#) <http://acm.sgu.ru/problem.php?contest=0&problem=458>

[Journal](#) <http://acm.sgu.ru/problem.php?contest=0&problem=494>

[More problems](#) http://codeforces.com/problemset/tags/dp/page/1?order=BY_SOLVED_DESC

กำหนดการพลวัต

Dynamic Programming

ผศ.ดร. ญัฐพงศ์ ชินชนะศ

นำมาจาก slides ของ รศ.ดร. สมชาย ประสิทธิ์ฐิตระกูล

หัวข้อ

- ❖ แบ่งใหญ่เป็นย่อย ถ้ามีปัญหาย่อยที่ซ้ำ จะซ้ำ
- ❖ จำคำตอบไว้ใช้ในอนาคต → เร็ว
- ❖ ใช้กับการแก้ปัญหาได้ทั้งแบบ
 - ❖ top-down
 - ❖ bottom-up
- ❖ กำหนดการพลวัต (Dynamic Programming)
- ❖ ตัวอย่าง

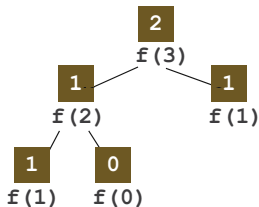
จำนวนฟีโบนัคชี (Fibonacci)

$$f_n = f_{n-1} + f_{n-2}$$

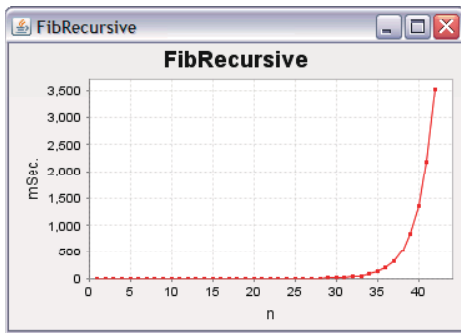
$$f_0 = 0, f_1 = 1$$

Top-down

```
f( n ) {  
    if (n < 2) return n  
    return f(n - 1) + f(n - 2)  
}
```



เวลาการทำงาน



Intel P8400 2.26GHz, Java 6u14

วิเคราะห์เวลาการทำงาน

```
f( n ) {
    if ( n < 2 ) return n
    return f( n - 1 ) + f( n - 2 )
}
```

ให้ $t(n)$ แทนจำนวนครั้งที่เรียก $f(n)$ จะได้

$$t(n) = t(n-1) + t(n-2) + 1, \quad t(0) = 1, \quad t(1) = 1$$

| | | | | | | | | | | |
|------|---|---|---|---|---|---|----|----|----|----|
| n | = | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| f(n) | = | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 |
| t(n) | = | 1 | 1 | 3 | 5 | 9 | 15 | 25 | 41 | 67 |

$$t(n) = 2f(n+1) - 1$$

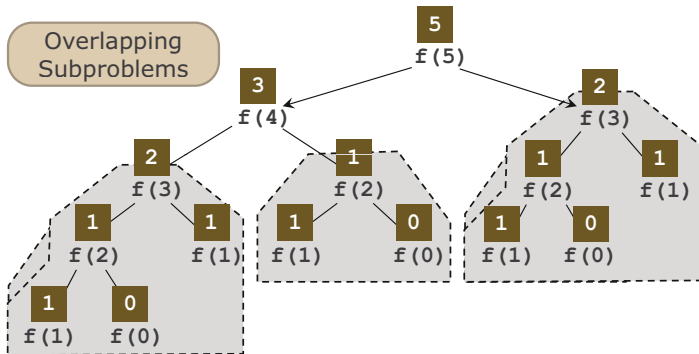
$$\Theta(\phi^n)$$

$$f(n) = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}, \quad \phi = \frac{1+\sqrt{5}}{2} \approx 1.61803...$$

ซ้ำเพราะคำนวณซ้ำ ๆ

```
f( n ) {  
    if ( n < 2 ) return n  
    return f( n - 1 ) + f( n - 2 )  
}
```

Overlapping
Subproblems



หลีกเลี่ยงการคำนวณซ้ำด้วยการจำ

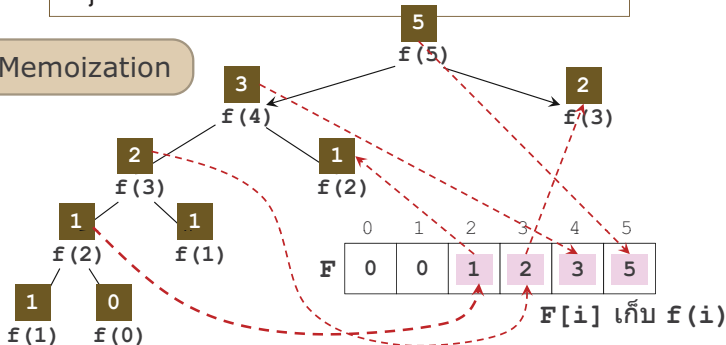
```

f(n, F[0..n]) {
  if (n < 2) return n
  if (F[n] > 0) return F[n]
  return F[n] = f(n-1, F) + f(n-2, F)
}

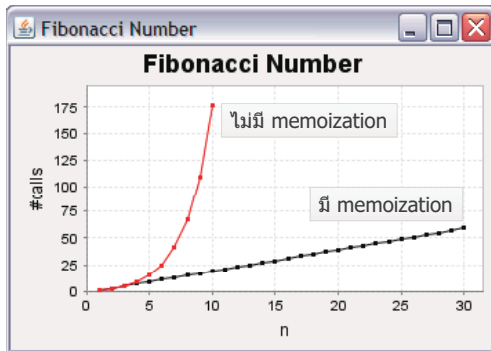
```

 $\Theta(n)$

Memoization



แบบมีกับไม่มี memoization



เติมตารางอีกแบบ : Bottom-Up

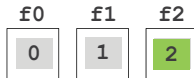
| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| F | 0 | 1 | 1 | 2 | 3 | 5 |

```
f( n ) {  
    F = new array[0..n]  
    F[0] = 0; F[1] = 1  
    for (i = 2; i <= n; i++) {  
        F[i] = F[i - 1] + F[i - 2]  
    }  
    return F[n]  
}
```

 $\Theta(n)$

bottom-up

ลดขนาดของตารางเหลือแค่ 3 ตัว



```
f( n ) {  
    f0 = 0, f1 = 1, f2 = 1  
    for (i = 2; i <= n; i++) {  
        f2 = f0 + f1  
        f0 = f1  
        f1 = f2  
    }  
    return f2  
}
```

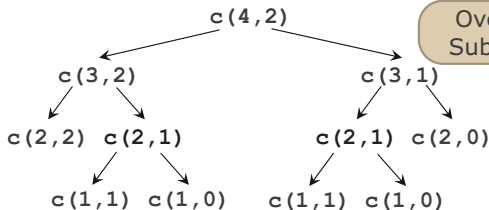
$C(n,k)$: เลือก k

$$C(n,k) = \begin{cases} C(n-1,k) + C(n-1,k-1) & \text{if } 0 < k < n \\ 1 & \text{if } k = 0 \text{ or } k = n \\ 0 & \text{otherwise} \end{cases}$$

```

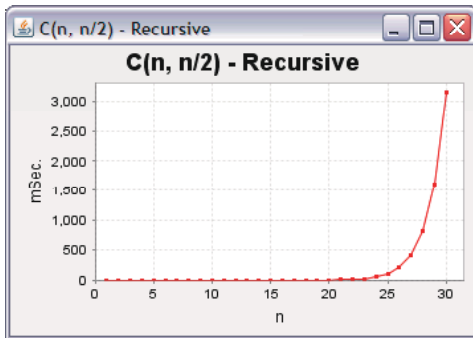
c(n, k) {
  if (k == 0 OR k == n) return 1
  if (k < 0 OR k > n) return 0
  return c(n-1, k) + c(n-1, k-1)
}

```



Overlapping
Subproblems

$C(n, k)$: เวลาการทำงาน



C(n,k) : Top-down + Memoization

c(10,5)

ทำ 503 ครั้ง

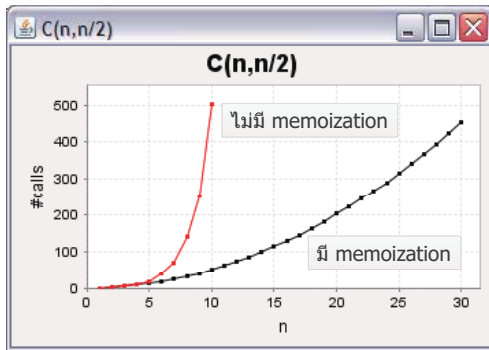
```
c(n, k) {  
    if (k == 0 OR k == n) return 1  
    if (k < 0 OR k > n) return 0  
    return c(n-1, k) + c(n-1, k-1)  
}
```

ทำ 51 ครั้ง

c(10, 5, new array[1..10][1..5])

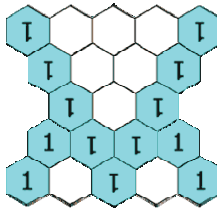
```
c(n, k, C[1..n][1..k]) {  
    if (k == 0 OR k == n) return 1  
    if (k < 0 OR k > n) return 0  
    if (C[n][k] > 0) return C[n][k]  
    return C[n][k] = c(n-1, k, C) + c(n-1, k-1, C)  
}
```

แบบมีกับไม่มี memoization



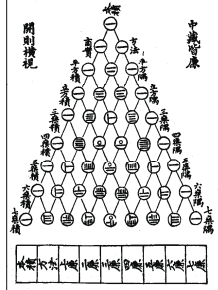
C(n, k) : Bottom-up

Pascal's Triangle



Bottom = หาคำตอบของกรณีเล็ก ๆ
 Up = หาคำตอบของกรณีใหญ่ขึ้น
 โดยใช้คำตอบของกรณีเล็กที่รู้แล้ว

圖方 察七 法 古



Pascal's Triangle : เขียนอีกแบบ

| n | k | | | | | | |
|---|---|---|----|----|----|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 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 |

$$C(6, 2) = 15$$

$C(n, k)$: Bottom-up

$$C(n, k) = \begin{cases} C(n-1, k) + C(n-1, k-1) & \text{if } 0 < k < n \\ 1 & \text{if } k = 0 \text{ or } k = n \\ 0 & \text{otherwise} \end{cases}$$

| | | k | | |
|---|---|---|---|----|
| | | 0 | 1 | 2 |
| n | 0 | 1 | | |
| | 1 | 1 | 1 | |
| | 2 | 1 | 2 | 1 |
| | 3 | 1 | 3 | 3 |
| | 4 | 1 | 4 | 6 |
| | 5 | 1 | 5 | 10 |
| | 6 | 1 | 6 | 15 |

ต้องการ $C(6, 2)$

C(n, k) : Bottom-up

```
c(n, k) {  
    C = new array[0..n][0..k]  
    for (i = 0; i <= n; i++) C[i][0] = 1  
    for (i = 0; i <= k; i++) C[i][i] = 1  
    for (i = 2; i <= n; i++)  
        for (j = 1; j <= k && j < i; j++)  
            C[i][j] = C[i - 1][j] + C[i - 1][j - 1]  
    return C[n][k];  
}
```

$$\Theta(nk)$$

| | k | | |
|---|---|---|----|
| | 0 | 1 | 2 |
| 0 | 1 | | |
| 1 | 1 | 1 | |
| 2 | 1 | 2 | 1 |
| 3 | 1 | 3 | 3 |
| 4 | 1 | 4 | 6 |
| 5 | 1 | 5 | 10 |
| 6 | 1 | 6 | 15 |

Top-Down vs. Bottom-Up

❖ Top-Down

- ❖ แบ่งปัญหาใหญ่เป็นปัญหาย่อย
- ❖ หาคำตอบของปัญหาย่อย
- ❖ นำคำตอบย่อย ๆ มารวมเป็นคำตอบของปัญหาใหญ่
- ❖ มักเขียนแบบ recursive
- ❖ แก้ปัญหาย่อยเท่าที่จำเป็น (แต่อาจแก้ซ้ำ แก้แล้วแก้อีก)
- ❖ ลดการแก้ปัญหาย่อยซ้ำด้วย memoization

❖ Bottom-Up

- ❖ เริ่มหาคำตอบของปัญหาเล็ก ๆ
- ❖ นำคำตอบของปัญหาเล็กมาหาคำตอบของปัญหาใหญ่ขึ้น
- ❖ มักเขียนแบบวงวน
- ❖ แก้ปัญหาย่อย ๆ ทุกปัญหา ปัญหาละครั้ง

Dynamic Programming

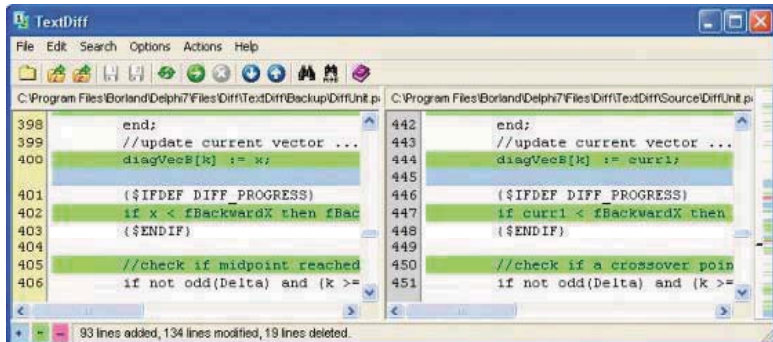
- ❖ ทำงานแบบ **bottom-up** ด้วยการแก้ปัญหาย่อยจำคำตอบที่ได้ เพื่อนำไปใช้แก้ปัญหาใหญ่ขึ้น
- ❖ มักใช้กับ **optimization problems**
 - ❖ การหาเส้นทางสั้นสุด
 - ❖ การหาลำดับย่อยรวมกันที่ยาวสุด
 - ❖ การเลือกของที่ได้มูลค่ารวมสูงสุด
 - ❖ การจัดเก็บข้อมูลในต้นไม้ค้นหาให้ค้นได้เร็วสุด
 - ❖ การหาลำดับการคูณเมทริกซ์ เพื่อให้คูณได้เร็วสุด
 - ❖ ...



Richard Bellman นักคณิตศาสตร์
ผู้คิดวิธี Dynamic programming
ในปี ค.ศ. 1953

Longest Common Subsequence

แฟ้มสองแฟ้มเหมือนกัน (ต่างกัน) ตรงไหนบ้าง



```
TextDiff
File Edit Search Options Actions Help

C:\Program Files\Borland\Delphi7\Files\Diff\TextDiff\Backup\DiffUnit.p  C:\Program Files\Borland\Delphi7\Files\Diff\TextDiff\Source\DiffUnit.p

398      end;
399      //update current vector ...
400      diagVecB[k] := x;
401      ($IFDEF DIFF_PROGRESS)
402      if x < fBackwardX then fBack
403      ($ENDIF)
404
405      //check if midpoint reached
406      if not odd(Delta) and (k >=

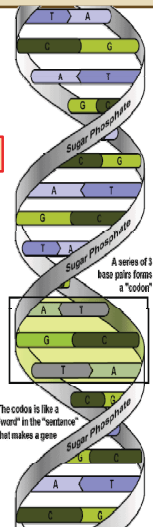
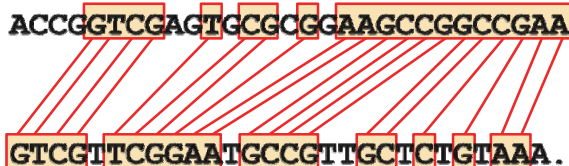
442      end;
443      //update current vector ...
444      diagVecB[k] := currl;
445
446      ($IFDEF DIFF_PROGRESS)
447      if currl < fBackwardX then
448      ($ENDIF)
449
450      //check if a crossover poin
451      if not odd(Delta) and (k >=

93 lines added, 134 lines modified, 19 lines deleted.
```

Longest Common Subsequence

วัดความคล้ายของ DNA sequences

ACCGGTCGAGTGCCTGAAGCCGGCCGAA
GTCGTTTCGGAATGCCGTTGCTCTGTAA.



Longest Common Subsequence

❖ $X = \langle H, E, L, L, O \rangle$ มี subsequences

❖ $\langle H, E, L, L, O \rangle \rightarrow \langle H, E \rangle$

❖ $\langle H, E, L, L, O \rangle \rightarrow \langle H, E, O \rangle$

❖ $\langle H, E, L, L, O \rangle \rightarrow \langle \rangle$

❖ ...

❖ $Y = \langle H, E, R, O \rangle$ มี subsequences

❖ $\langle H, E, R, O \rangle \rightarrow \langle H, E \rangle$

❖ $\langle H, E, R, O \rangle \rightarrow \langle H, R \rangle$

❖ $\langle H, E, R, O \rangle \rightarrow \langle H, E, O \rangle$

❖ ...

❖ common subsequence(X, Y)

❖ longest common subsequence(X, Y)

นิยาม

$$\diamond X = \langle x_1, x_2, x_3, \dots, x_m \rangle$$

$$\diamond Y = \langle y_1, y_2, y_3, \dots, y_n \rangle$$

$$\diamond X_i = \langle x_1, x_2, \dots, x_i \rangle$$

$$\diamond Y_j = \langle y_1, y_2, \dots, y_j \rangle$$

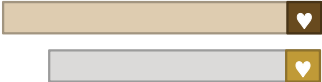
$$\diamond \text{LCS}(X_i, Y_j) :$$

longest common subsequence ของ X_i, Y_j

$$\diamond \text{LCS}(X, Y) = \text{LCS}(X_m, Y_n)$$

$$\diamond L(i, j) : \text{ความยาวของ } \text{LCS}(X_i, Y_j)$$

คำตอบใหญ่ได้จากคำตอบย่อย ๆ

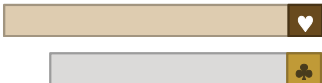
LCS()

=

LCS() + 

LCS(HELL**O**, HER**O**) = LCS(**HELL**, **HER**) + **O**
 = **HE** + **O**
 = **HEO**

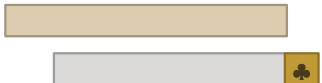
คำตอบใหญ่ได้จากคำตอบย่อย ๆ

LCS() LCS(COP, CEO)

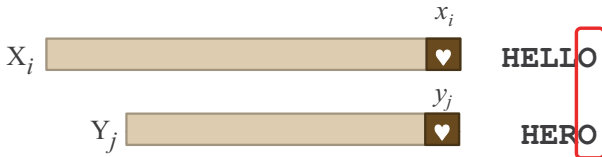
=

=

LCS() LCS(**C**OP, **C**E)

LCS() LCS(**CO**, **CEO**)

Recurrence ของ $L(i, j)$

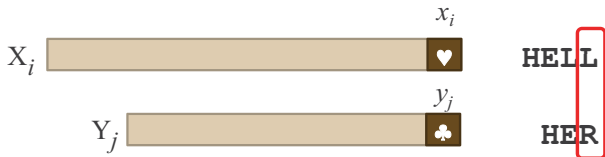


$$L(i, j) = L(i - 1, j - 1) + 1 \quad \text{if } x_i = y_j$$

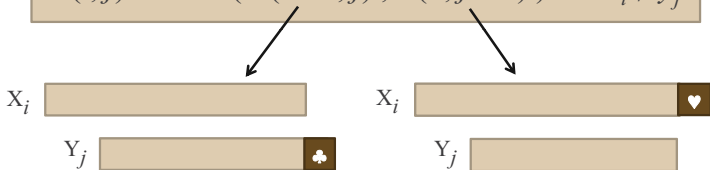
↑
ความยาวของ $LCS(X_{i-1}, Y_{j-1})$

↑
ตัวซ้ำที่เท่ากัน 1 ตัว

Recurrence ของ $L(i, j)$



$$L(i, j) = \max(L(i-1, j), L(i, j-1)) \quad \text{if } x_i \neq y_j$$



Recurrence ของ $L(i, j)$

X_i x_i

Y_j y_j

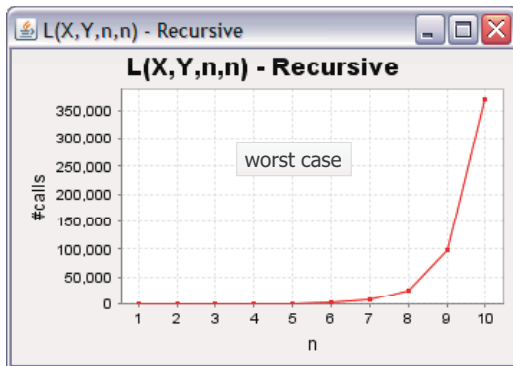
$$L(i, j) = \begin{cases} L(i-1, j-1) + 1 & \text{if } x_i = y_j \\ \max(L(i-1, j), L(i, j-1)) & \text{if } x_i \neq y_j \\ 0 & \text{if } i = 0 \text{ or } j = 0 \end{cases}$$

LCS : Top-down

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

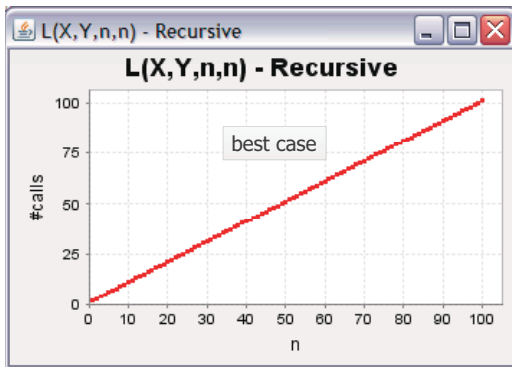
```
L(x[1..m], y[1..n]) {  
    return L(x, y, m, n)  
}  
L(x[1..m], y[1..n], i, j) {  
    if (i == 0 OR j == 0) return 0  
    if (x[i] == y[j]) {  
        return L(x, y, i-1, j-1) + 1  
    } else {  
        return max( L(x, y, i-1, j),  
                    L(x, y, i, j-1) )  
    }  
}
```


หาความยาว LCS : เวลาการทำงาน



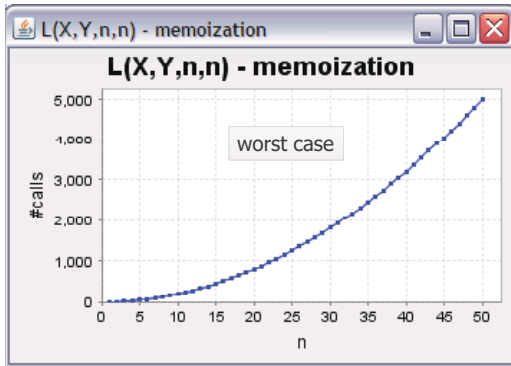
ลักษณะข้อมูลของกรณี worst case และ best case เป็นอย่างไร ?
กรณี best case มีประสิทธิภาพเชิงเวลาอย่างไร ?
ลองปรับด้วย memoization จะได้ผลอย่างไร ?

LCS : best cases



worst case เมื่อสองลำดับไม่เหมือนกันเลย
best case เมื่อสองลำดับเหมือนกันหมด

LCS : memoization



LCS : Bottom-up

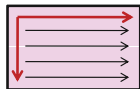
$$L(i, j) = \begin{cases} L(i-1, j-1) + 1 & \text{if } x_i = y_j \\ \max(L(i-1, j), L(i, j-1)) & \text{if } x_i \neq y_j \\ 0 & \text{if } i = 0 \text{ or } j = 0 \end{cases}$$

| i \ j | | H | E | L | L | O |
|-------|--|---|---|---|---|---|
| | | 0 | 0 | 0 | 0 | 0 |
| H | | 0 | 1 | 1 | 1 | 1 |
| E | | 0 | 1 | 2 | 2 | 2 |
| R | | 0 | 1 | 2 | 2 | 2 |
| O | | 0 | 1 | 2 | 2 | 3 |

LCS : Dynamic Prog.

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

```
LCS_Length(x[1..m], y[1..n]) {  
    L = new array[0..m][0..n]  
    for (i = 0; i <= m; i++) L[i][0] = 0  
    for (j = 0; j <= n; j++) L[0][j] = 0  
    for (i = 1; i <= m; i++)  
        for (j = 1; j <= n; j++)  
            if (x[i] == y[j])  
                L[i][j] = L[i - 1][j - 1] + 1  
            else  
                L[i][j] = max(L[i - 1][j], L[i][j - 1])  
    return L;  
}
```



$\Theta(nm)$

ต้องการ LCS : จำผลการตัดสินใจ

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

| j | | H E L L O | | | | | |
|---|---|-----------|---|---|---|---|---|
| i | | 0 | 0 | 0 | 0 | 0 | 0 |
| | H | 0 | 1 | 1 | 1 | 1 | 1 |
| | E | 0 | 1 | 2 | 2 | 2 | 2 |
| | R | 0 | 1 | 2 | 2 | 2 | 2 |
| | O | 0 | 1 | 2 | 2 | 2 | 3 |

จำผลการตัดสินใจ

```
LCS(x[1..m], y[1..n]) {  
    L = new array[0..m][0..n]  
    D = new array[1..m][1..n]  
    for (i = 0; i <= m; i++) L[i][0] = 0  
    for (j = 0; j <= n; j++) L[0][j] = 0  
    for (i = 1; i <= m; i++)  
        for (j = 1; j <= n; j++)  
            if (x[i] == y[j])  
                L[i][j] = L[i - 1][j - 1] + 1  
                D[i][j] = ↖  
            else  
                if (L[i - 1][j] > L[i][j - 1])  
                    L[i][j] = L[i - 1][j]  
                    D[i][j] = ↑  
                else  
                    L[i][j] = L[i][j - 1]  
                    D[i][j] = ←
```


ใช้ผลการตัดสินใจสร้างคำตอบ

```

lcs = an empty sequence
i = n, j = m
while (i > 0 AND j > 0) {
    switch( D[i][j] ) {
        case : ↖
            lcs = x[i] + lcs
            i--; j--
        case : ←
            j--
        case : ↑
            i--
    }
}
return lcs
}

```

| | | j | | | | |
|---|---|---|---|---|---|---|
| | | H | E | L | L | O |
| i | | | | | | |
| | H | ↖ | ← | ← | ← | ← |
| | E | ↑ | ↖ | ← | ← | ← |
| | R | ↑ | ↑ | ← | ← | ← |
| | O | ↑ | ↑ | ← | ← | ↖ |

ต้องการประหยัด ไม่จำผลการตัดสินใจ

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

| | | j | | | | | |
|---|--|---|---|---|---|---|--|
| i | | H | E | L | L | O | |
| | | 0 | 0 | 0 | 0 | 0 | |
| H | | 0 | 1 | 1 | 1 | 1 | |
| E | | 0 | 1 | 2 | 2 | 2 | |
| R | | 0 | 1 | 2 | 2 | 2 | |
| O | | 0 | 1 | 2 | 2 | 3 | |

สร้าง LCS ได้จากตาราง L

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

| | | j | | | | | |
|---|---|---|---|---|---|---|--|
| | | H | E | L | L | O | |
| i | | 0 | 0 | 0 | 0 | 0 | |
| | H | 0 | 1 | 1 | 1 | 1 | |
| | E | 0 | 1 | 2 | 2 | 2 | |
| | R | 0 | 1 | 2 | 2 | 2 | |
| | O | 0 | 1 | 2 | 2 | 3 | |

สร้าง LCS ได้จากตาราง L

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

| j | | | | | | | |
|-----|--|---|---|---|---|---|--|
| i \ | | H | E | L | L | O | |
| | | 0 | 0 | 0 | 0 | 0 | |
| H | | 0 | 1 | 1 | 1 | 1 | |
| E | | 0 | 1 | 2 | 2 | 2 | |
| R | | 0 | 1 | 2 | 2 | 2 | |
| O | | 0 | 1 | 2 | 2 | 3 | |

สร้าง LCS ได้จากตาราง L

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

| j i \ | | H | E | L | L | O |
|----------|--|---|---|---|---|---|
| | | 0 | 0 | 0 | 0 | 0 |
| H | | 0 | 1 | 1 | 1 | 1 |
| E | | 0 | 1 | 2 | 2 | 2 |
| R | | 0 | 1 | 2 | 2 | 2 |
| O | | 0 | 1 | 2 | 2 | 3 |

สร้าง LCS ได้จากตาราง L

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

| | | | | | | | |
|--|---|---|---|---|---|---|---|
| | | j | | | | | |
| | i | | H | E | L | L | O |
| | | 0 | 0 | 0 | 0 | 0 | 0 |
| | H | 0 | 1 | 1 | 1 | 1 | 1 |
| | E | 0 | 1 | 2 | 2 | 2 | 2 |
| | R | 0 | 1 | 2 | 2 | 2 | 2 |
| | O | 0 | 1 | 2 | 2 | 2 | 3 |

สร้าง LCS ได้จากตาราง L

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

| | <div>H</div> | <div>E</div> | L | L | <div>O</div> |
|--------------|--------------|--------------|---|---|--------------|
| <div>H</div> | 0 | 0 | 0 | 0 | 0 |
| <div>E</div> | 0 | 1 | 1 | 1 | 1 |
| <div>R</div> | 0 | 1 | 2 | 2 | 2 |
| <div>O</div> | 0 | 1 | 2 | 2 | 3 |

สร้าง LCS ได้จากตาราง L

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

| | | | | | | |
|-----|---|---|---|---|---|---|
| | | | | | | |
| | j | | | | | |
| i \ | | H | E | L | L | O |
| | 0 | 0 | 0 | 0 | 0 | 0 |
| H | 0 | 1 | 1 | 1 | 1 | 1 |
| E | 0 | 1 | 2 | 2 | 2 | 2 |
| R | 0 | 1 | 2 | 2 | 2 | 2 |
| O | 0 | 1 | 2 | 2 | 2 | 3 |

LCS_Soln(x, y, L)

```

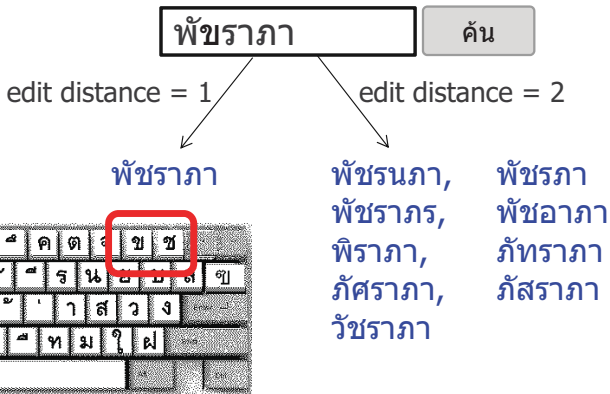
LCS_Soln(x[1..n], y[1..m], L[0..n][0..m]) {
    lcs = an empty sequence
    i = n, j = m
    while (i > 0 AND j > 0) {
        if (x[i] == y[j]) {
            lcs = x[i] + lcs
            i--; j--
        } else {
            if (L[i][j-1] > L[i-1][j])
                j--
            else
                i--
        }
    }
    return lcs
}

```

สามารถหา LCS ได้ด้วยเนื้อที่ $O(n + m)$

D. Hirschberg "A linear space algorithm for computing maximal common subsequences"

สะกดผิด : หาคำใกล้เคียง



Minimum Edit Distance

- ❖ เปลี่ยน HELLO ให้เป็น HERO เช่น
 - ❖ HELLO \rightarrow ELLO \rightarrow LLO \rightarrow LO \rightarrow RO \rightarrow ERO \rightarrow HERO
 - ❖ HELLO \rightarrow HELO \rightarrow HERO
- ❖ ให้การแก้ไขสตริงประกอบด้วย
 - ❖ การลบอักขระหนึ่งตัว
 - ❖ การเพิ่มอักขระหนึ่งตัว
 - ❖ การแทนอักขระหนึ่งตัวด้วยอักขระอีกตัว
- ❖ การแก้ไขมีต้นทุน c_D , c_I , c_S
- ❖ Edit distance คือต้นทุนการแก้ไขเพื่อเปลี่ยนสตริง
- ❖ ปัญหา : เปลี่ยนสตริง s ให้เป็น t อย่างไรให้มี minimum edit distance

ลองทำเอง
คล้าย ๆ LCS

ลักษณะของปัญหา

❖ overlapping subproblems :

- ❖ ต้องแก้ปัญหาย่อย ๆ ซ้ำ ๆ หลายครั้ง
การจำคำตอบจึงประหยัดเวลาการทำงานได้มาก
- ❖ ต้องมีจำนวนปัญหาย่อยทั้งหมดไม่มาก

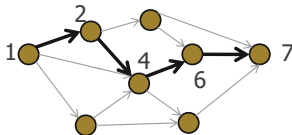
❖ optimal substructures (principle of optimality) :

- ❖ คำตอบที่ดีที่สุดของปัญหาใหญ่ได้มาจาก
คำตอบที่ดีที่สุดของปัญหาที่เล็กกว่า
- ❖ ทำให้เขียน recurrence ของคุณภาพของคำตอบได้
- ❖ คิด : ใช้คำตอบที่ดีที่สุดของปัญหาเล็กใด และเลือกอย่างไร

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

Optimal Substructures

LCS("HELLO", "HERO") คือ LCS("HELL", "HER") ต่อด้วย "O"



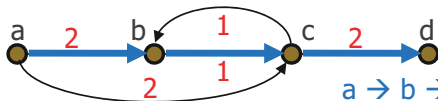
ถ้าทางเดินสั้นสุดจาก 1 ไป 7 คือ $1 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 7$
แสดงว่า

| | | |
|---------------|---|-----------------|
| ทางจาก 1 ไป 4 | ซึ่งคือ $1 \rightarrow 2 \rightarrow 4$ | ต้องสั้นสุดด้วย |
| ทางจาก 1 ไป 6 | ซึ่งคือ $1 \rightarrow 2 \rightarrow 4 \rightarrow 6$ | ต้องสั้นสุดด้วย |
| ทางจาก 2 ไป 6 | ซึ่งคือ $2 \rightarrow 4 \rightarrow 6$ | ต้องสั้นสุดด้วย |
| ทางจาก 2 ไป 7 | ซึ่งคือ $2 \rightarrow 4 \rightarrow 6 \rightarrow 7$ | ต้องสั้นสุดด้วย |

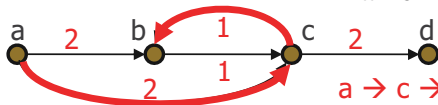
ปัญหา optimization ไม่จำเป็นต้องมี
optimal substructure เสมอไป

Longest Simple Path

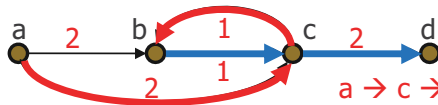
Simple path คือ path ที่ไม่ผ่านปมซ้ำ



$a \rightarrow b \rightarrow c \rightarrow d$ ยาวสุด
แต่ $a \rightarrow b$ ไม่ยาวสุด



$a \rightarrow c \rightarrow b$ ยาวกว่า

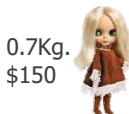


$a \rightarrow c \rightarrow b \rightarrow c \rightarrow d$
ยาวกว่า แต่ไม่ใช่ simple path

ไม่มี optimal substructure
ใช้ dynamic programming ไม่ได้

0/1 Knapsack

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

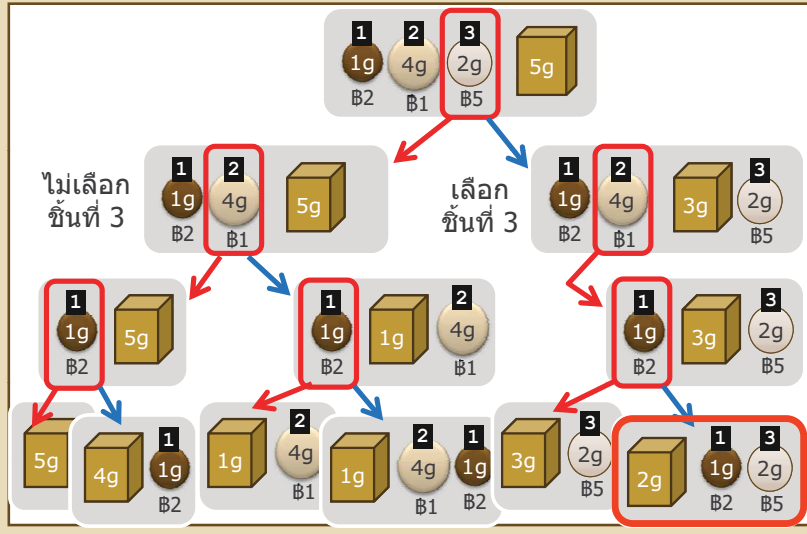


0/1 Knapsack

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

$$\begin{aligned} & \text{maximize} \quad \sum_{k=1}^n x_k v_k \\ & \text{subject to} \quad \sum_{k=1}^n x_k w_k \leq W \\ & \quad \quad \quad x_k \in \{0, 1\} \end{aligned}$$

แบ่งปัญหาใหญ่เป็นปัญหาย่อย



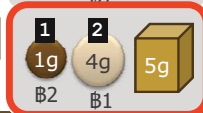
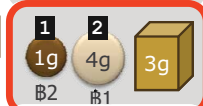
หาคำตอบใหญ่จากคำตอบเล็ก



$$\max(\text{B3}, \text{B5} + \text{B2})$$

เลือกชั้นที่ 3

ไม่เลือกชั้นที่ 3

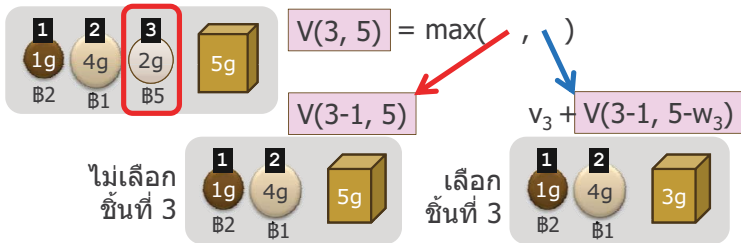


ปัญหาใหญ่ : ปัญหาย่อย

- ❖ **LCS** : $L(i, j)$
 - ❖ ปัญหาใหญ่หรือเล็กขึ้นกับความยาวของลำดับ X_i และ Y_j
 - ❖ **Edit distance** : $D(i, j)$
 - ❖ **Knapsack** : $V(?)$
 - ❖ ปัญหาใหญ่หรือเล็กขึ้นกับจำนวนของ และน้ำหนักที่ถูกรับได้
- $V(i, j)$: มูลค่ารวมสูงสุดในการ
- ❖ เลือกของชิ้นที่ 1, 2, 3, ..., i
 - ❖ ใส่ลงเป้ที่รับน้ำหนักได้ไม่เกิน j

เขียน recurrence ของ $V(i, j)$

เขียน recurrence $V(i, j)$



$V(i, j)$ แทนมูลค่าของการเลือกที่ดีที่สุดเมื่อ
มีของให้เลือกชั้นที่ 1, 2, ..., i ใส่ถุงเป้ที่จุได้นัก j

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

Knapsack : Top-down

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

```

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] )
    }
}

```

Top-down + Memoization

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

```
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]) )  
    }  
}
```

0/1 Knapsack : Bottom-Up

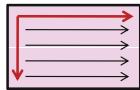
$$V(i, j) = \begin{cases} \max(V(i-1, j), v_i + V(i-1, j - w_i)) & \text{if } i > 0 \text{ and } j \geq w_i \\ V(i-1, j) & \text{if } j < w_i \\ 0 & \text{if } i = 0 \text{ or } j = 0 \end{cases}$$

| | | j | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------|-------|---|---|---|----|----|----|----|----|-----|-----|-----|-----|
| v_i | w_i | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 1 | 0 | 0 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| | | 2 | 0 | 0 | 30 | 30 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| | | 3 | 0 | 0 | 30 | 66 | 66 | 96 | 96 | 116 | 116 | 116 | 116 |
| | | 4 | 0 | 0 | 30 | 66 | 66 | 96 | 96 | 116 | 116 | 136 | 136 |
| | | 5 | 0 | 0 | 30 | 66 | 66 | 96 | 96 | 116 | 126 | 136 | 156 |

$W = 10$

0/1 Knapsack : 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  
}
```



$\Theta(nW)$

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

อยากรู้เลือกอะไร : จำผลการตัดสินใจ

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

| | | j | | | | | | |
|-------|-------|---|---|-----|------|------|------|------|
| | | i | 0 | 1 | 2 | 3 | 4 | 5 |
| v_i | w_i | 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 ✓ |

$$W = 5$$

ใช้ผลการตัดสินใจสร้างคำตอบ

| | | | j | | 0 | 1 | 2 | 3 | 4 | 5 |
|-------|-------|---|---|---|---|---|---|---|---|---|
| v_i | w_i | i | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 1 | 0 | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 30 | 2 | 2 | 0 | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 25 | 3 | 3 | 0 | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |

O/1 Knapsack : จำการตัดสินใจ

```

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;

```

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

ต้องการประหยัด ไม่จำผลการตัดสินใจ

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

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

$i = 0 \text{ or } j = 0 ?$

→ false

$j < w_i ?$

→ false

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

แล้วเลือกของชิ้นใดบ้าง ?

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

| | | | j | | | | | | | | | | | |
|-------|-------|---|---|---|---|----|----|----|----|----|-----|-----|-----|-----|
| | | | i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| v_i | w_i | 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 |

$i = 0 \text{ or } j = 0$?

→ false

$j < w_i$?

→ false

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

→ false

แล้วเลือกของชิ้นใดบ้าง ?

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

| | | | j | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------|-------|---|---|---|---|----|----|----|----|----|-----|-----|-----|-----|
| | | i | | | | | | | | | | | | |
| v_i | w_i | 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 |

$i = 0 \text{ or } j = 0 ?$

→ false

$j < w_i ?$

→ false

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

แล้วเลือกของชิ้นใดบ้าง ?

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

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

$i = 0 \text{ or } j = 0 ?$

→ false

$j < w_i ?$

→ false

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

แล้วเลือกของชิ้นใดบ้าง ?

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

| | | j | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------|-------|---|---|---|----|----|----|----|----|-----|-----|-----|-----|
| v_i | w_i | i | 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 $j = 0$?

→ true

knapsack_Soln(v, w, V)

```
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;  
}
```

ทอนเงิน

- ❖ มีเหรียญอยู่ n แบบ : $1, 2, 3, \dots, n$
- ❖ แต่ละแบบมีค่า : $v_1, v_2, v_3, \dots, v_n$
- ❖ ต้องการทอนเงินให้ลูกค้ามูลค่า V
- ❖ โดยใช้เหรียญเป็นจำนวนน้อยสุด
 - ❖ (มีเหรียญแต่ละแบบจำนวนไม่จำกัด)
- ❖ ตัวอย่าง
 - ❖ มีเหรียญ : $1, 3, 4, 10$
 - ❖ ต้องการทอนเงินมูลค่า 6
 - ❖ คำตอบ : $3, 3$

ลองทำเอง
คล้าย ๆ Knapsack

Sum of Subset

❖ กำหนดให้

- ❖ S เป็นเซตของจำนวน
- ❖ K เป็นจำนวน

ลองทำเอง
คล้าย ๆ Knapsack

❖ จงหาว่า

- ❖ มีเซตย่อยของ S ที่ผลรวมของจำนวนมีค่าเท่ากับ K หรือไม่

❖ ตัวอย่าง

- ❖ $S = \{1, 3, 2, 9, 8\}$, $K = 11 \rightarrow$ คำตอบ : มี
- ❖ $S = \{1, 3, 2, 9, 8\}$, $K = 7 \rightarrow$ คำตอบ : ไม่มี

Maximum Contiguous Sum

❖ กำหนดให้

❖ A คือลำดับของจำนวน $\langle a_1, a_2, \dots, a_n \rangle$

❖ จงหา

❖ ช่วงของข้อมูลที่ติดกันใน A ที่ผลรวมมีค่ามากที่สุด

❖ ตัวอย่าง

❖ $A = \langle 31, -41, 59, 26, -53, 58, 97, -93, -23, 84 \rangle$

❖ แนวคิด

❖ ลุยทุกช่วงของข้อมูล มีทั้งหมด $C(n,2)$ ช่วง ใช้เวลา $\Theta(n^2)$

❖ ใช้การแบ่งแยกและเอาชนะ

ลองทำเอง ❖ แบ่งครึ่ง, หาฝั่งซ้าย, หาฝั่งขวา, หาข้ามฝั่ง

❖ ใช้เวลา $\Theta(n \log n)$

Dynamic Programming

❖ เขียน recurrence ของผลรวม

- ❖ $P(i)$ คือผลรวมของข้อมูลที่ติดกันใน A ที่มีค่ามากที่สุด โดยให้ตัวท้ายสุดอยู่ที่ตำแหน่ง i

$$P(i) = \max(P(i-1) + a_i, a_i) \quad P(1) = a_1$$

- ❖ $S(i)$ คือผลรวมของข้อมูลที่ติดกันใน $A[1..i]$ ที่มีค่ามากที่สุด (คำตอบก็คือ $S(n)$)

$$S(i) = \max_{1 \leq k \leq i} \{ P(k) \} = \max(\max_{1 \leq k \leq i-1} \{ P(k) \}, P(i))$$

$$S(i) = \max(S(i-1), P(i)) \quad S(1) = a_1$$

$$A = \langle 2, 1, -4, 2 \rangle \rightarrow P(1) = 2, P(2) = 3, P(3) = -1, P(4) = 2 \\ S(1) = 2, S(2) = 3, S(3) = 3, S(4) = 3$$

Dynamic Programming

```

MCS_Value(A[1..n]) {
    create S[1..n], P[1..n];
    P[1] = A[1];
    for (i = 2; i <= n; i++)
        P[i] = max(P[i - 1] + A[i], A[i]);
    S[1] = A[1];
    for (i = 2; i <= n; i++)
        S[i] = max(S[i - 1], P[i]);
    return S;
}

```

$$P(i) = \max(P(i-1) + a_i, a_i)$$

$$S(i) = \max(S(i-1), P(i))$$

อย่างไร ถ้าต้องการ
หาช่วงของ A ที่ผลรวมมี
ค่ามากที่สุด

```

MCS_Value(A[1..n]) {
    P = A[1]; S = A[1];
    for (i = 2; i <= n; i++) {
        P = max(P + A[i], A[i]);
        S = max(S, P);
    }
    return S;
}

```

$$\Theta(n)$$

ตัดข้อความเป็นคำ ๆ

- ❖ **Input** : ข้อความภาษาไทย
- ❖ **Output** : รายการสิ้นสุดของคำที่ตัดจากข้อความ

นายกนกรกรนดั่ง



นาย กนก กรน ดั่ง

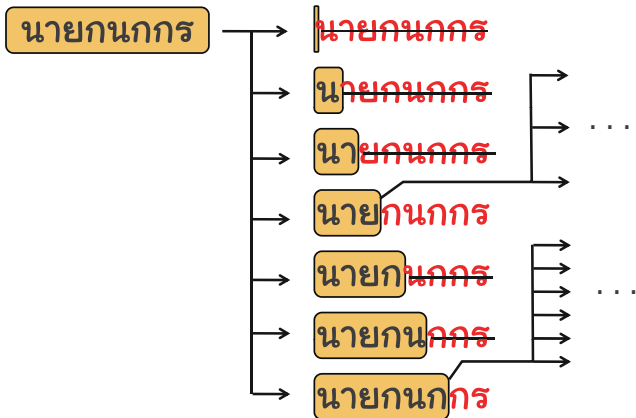
นายกนกรกร



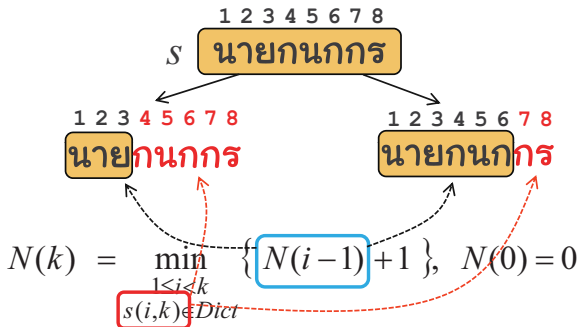
นาย กนกกร

แบ่งคำโดยใช้พจนานุกรม

ตัดข้อความเป็นคำ ๆ : Top-Down



ตัดข้อความเป็นคำ ๆ : Top-Down



ให้ s แทนสตริงของข้อความที่ได้รับ

$s(i, k)$ แทนสตริงย่อยตั้งแต่ตัวที่ i ถึง k

$N(k)$ แทนจำนวนคำน้อยสุดที่แบ่งได้จาก $s(1, k)$

ตัดข้อความเป็นคำ ๆ : Top-Down

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

```
N( dict, s[1..n], k ) {  
  if (k == 0) return 0  
  minN = ∞  
  for (i = 1; i < k; i++) {  
    if ( s[i..k] ∈ dict ) {  
      minN = min( minN, 1 + N( dict, s, i-1 ) )  
    }  
  }  
  return minN  
}
```

จำนวนค่าน้อยสุดที่
แบ่งจากข้อความ

ต้องการรายการของคำ

```
WordSep( dict, s[1..n], k ) {  
    if (k == 0) return an empty list  
    minN = ∞  
    minWords = null  
    for (i = 1; i < k; i++) {  
        if ( s[i..k] ∈ dict ) {  
            words = WordSep( dict, s, i-1 )  
            if (words ≠ null) {  
                words.add( s[i..k] )  
                if ( words.size() < minN ) {  
                    minN = words.size()  
                    minWords = words  
                }  
            }  
        }  
    }  
    return minWords  
}
```

Overlapping Subproblems
(ลองยกตัวอย่าง)

ตัดข้อความเป็นคำ ๆ : Bottom-Up

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

$k = 2$

$i=1$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|

 น ำ ย ัก ัก ัก ัก

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|

 น ำ ย ัก ัก ัก ัก

ตัดข้อความเป็นคำ ๆ : Bottom-Up

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

 $k = 3$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

$i=1$ น ำ ย ก น ก ก ร

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ร

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | - | - | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

$i=2$ น ำ ย ก น ก ก ร

ตัดข้อความเป็นคำ ๆ : Bottom-Up

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

$k = 4$

$i=1$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ัก น ัก ัก ร

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ัก น ัก ัก ร

$i=2$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ~~ย ัก~~ น ัก ัก ร

$i=3$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 2 | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ัก น ัก ัก ร

$k = 5$

$i=4$ น า ย **ก น** ก ก ร

น ำ ย ก น ก ร

ตัดข้อความเป็นคำ ๆ : Bottom-Up

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

$k = 6$

$i=1$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | - | - | - |
| | | น | า | ย | ก | น | ก | |

การตัดที่ถูกต้อง: น าย ก น ก

$i=2$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | - | - | - |
| | | น | า | ย | ก | น | ก | |

การตัดที่ถูกต้อง: น าย ก น ก

$i=3$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | - | - | - |
| | | น | า | ย | ก | น | ก | |

การตัดที่ถูกต้อง: น าย ก น ก

$i=4$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
| | | น | า | ย | ก | น | ก | |

การตัดที่ถูกต้อง: น าย ก น ก

$i=5$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
| | | น | า | ย | ก | น | ก | |

การตัดที่ถูกต้อง: น าย ก น ก

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
| | | น | า | ย | ก | น | ก | |

การตัดที่ถูกต้อง: น าย ก น ก

ตัดข้อความเป็นคำ ๆ : Bottom-Up

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

$k = 7$

$i=1$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ร

$i=2$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ร

$i=3$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ร

$i=4$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ร

$i=5$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ร

$i=6$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ก ร

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ก ร

ตัดข้อความเป็นคำ ๆ : Bottom-Up

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

$k = 8$

$i=1$ 0 - 1 1 1 2 2 3 -

น ำ ย ก ~~น~~ ก ก ร

$i=2$ 0 - 1 1 1 2 2 3 -

น ำ ย ก ~~น~~ ก ก ร

$i=3$ 0 - 1 1 1 2 2 3 -

น ำ ย ก ~~น~~ ก ก ร

$i=4$ 0 - 1 1 2 2 3 2

น ำ ย ก น ก ก ร

$i=5$ 0 - 1 1 1 2 2 3 -

น ำ ย ก ~~น~~ ก ก ร

$i=6$ 0 - 1 1 1 2 2 3 -

น ำ ย ก น ก ~~ก~~ ร

$i=7$ 0 - 1 1 1 2 2 3 3

น ำ ย ก น ก ก ร

0 - 1 1 1 2 2 3 2

น ำ ย ก น ก ก ร

ตัดข้อความเป็นคำ ๆ : Bottom-Up

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

```

N( dict, s[1..n] ) {
  N = new array[0..n]
  N[0] = 0; N[1..n] = ∞
  for( k = 2; k <= n; k++ ) {
    for( i = 1; i < k; i++ ) {
      if ( s[i..k] ∈ dict ) {
        N[k] = min( N[k], N[i-1]+1 )
      }
    }
  }
  return N
}

```

ถ้าการค้นในพจนานุกรม
ใช้เวลา $\Theta(1)$

$\Theta(n^2)$

จำนวนค่าน้อยสุดที่
แบ่งจากข้อความ

ต้องการรายการของคำ

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

ที่แต่ละค่าของ k

ต้องจำว่า i ใดที่ได้ค่า $N(i-1) + 1$ น้อยสุด

จำค่า i ที่ได้ min

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in Dict}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

 $k = 2$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|

 $i = 1$ น ำ ย ก น ก ก ร

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ร

จำค่า i ที่ได้ min

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

$k = 3$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

 $i = 1$ น ำ ย ก น ก ก ร

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

น ำ ย ก น ก ก ร

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | - | - | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

 $i = 2$ น ำ ย ก น ก ก ร

จำค่า i ที่ได้ min

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

 $k = 4$

$i=1$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

 น ำ ย ก น ก ก ร

$i=2$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | - | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

 น ~~ำ~~ ~~ย~~ ก น ก ก ร

$i=3$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 2 | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

 น ำ **ย** **ก** น ก ก ร

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | - | - | - | - |
|---|---|---|---|---|---|---|---|---|

 น ำ ย ก น ก ก ร

จำค่า i ที่ได้ min

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

$k = 5$

$i=1$ 0 - 1 1 1 - - - -

น ำ ย ก น ก ก ร

$i=2$ 0 - 1 1 1 - - - -

น ำ ย ก น ก ก ร

$i=3$ 0 - 1 1 1 - - - -

น ำ ย ก น ก ก ร

$i=4$ 0 - 1 1 1 2 - - - -

น ำ ย ก น ก ก ร

0 - 1 1 1 2 - - - -

น ำ ย ก น ก ก ร

จำค่า i ที่ได้ min

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

 $k = 6$

| | | | | | | | | | |
|-------|---|---|---|--------------|---|---|---|---|---|
| $i=1$ | 0 | - | 1 | 1 | 1 | 2 | - | - | - |
| | น | า | ย | ก | น | ก | ก | ร | |
| $i=2$ | 0 | - | 1 | 1 | 1 | 2 | - | - | - |
| | น | า | ย | ก | น | ก | ก | ร | |
| $i=3$ | 0 | - | 1 | 1 | 1 | 2 | - | - | - |
| | น | า | ย | ก | น | ก | ก | ร | |
| $i=4$ | 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
| | น | า | ย | ก | น | ก | ก | ร | |
| $i=5$ | 0 | - | 1 | 1 | 1 | 2 | 2 | - | - |
| | น | า | ย | ก | น | ก | ก | ร | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|
| | | | 1 | 1 | 1 | 4 | | | |
| 0 | - | 1 | 1 | 1 | 2 | 2 | - | - | |
| | น | า | ย | ก | น | ก | ก | ร | |

จำค่า i ที่ได้ min

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

$k = 7$

$i=1$ 0 - 1 1 1 2 2 - -

น ำ ย ก น ก ร

0 - 1 1 1 2 2 - -

$i=2$ น ำ ย ก น ก ร

0 - 1 1 1 2 2 - -

$i=3$ น ำ ย ก น ก ร

0 - 1 1 1 2 2 - -

$i=4$ น ำ ย ก น ก ร

0 - 1 1 1 2 2 - -

$i=5$ น ำ ย ก น ก ร

0 - 1 1 1 2 2 3 -

$i=6$ น ำ ย ก น ก ร

0 - 1 1 1 2 2 3 -

น ำ ย ก น ก ร

จำค่า i ที่ได้ min

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

 $k = 8$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | - |
|---|---|---|---|---|---|---|---|---|

 $i=1$

| | | | | | | |
|---|---|---|---|---|---|---|
| น | า | ย | ก | น | ก | ร |
|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | - |
|---|---|---|---|---|---|---|---|---|

 $i=2$

| | | | | | | |
|---|---|---|---|---|---|---|
| น | า | ย | ก | น | ก | ร |
|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | - |
|---|---|---|---|---|---|---|---|---|

 $i=3$

| | | | | | | |
|---|---|---|---|---|---|---|
| น | า | ย | ก | น | ก | ร |
|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|

 $i=4$

| | | | | | | |
|---|---|---|---|---|---|---|
| น | า | ย | ก | น | ก | ร |
|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | - |
|---|---|---|---|---|---|---|---|---|

 $i=5$

| | | | | | | |
|---|---|---|---|---|---|---|
| น | า | ย | ก | น | ก | ร |
|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | - |
|---|---|---|---|---|---|---|---|---|

 $i=6$

| | | | | | | |
|---|---|---|---|---|---|---|
| น | า | ย | ก | น | ก | ร |
|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | 3 |
|---|---|---|---|---|---|---|---|---|

 $i=7$

| | | | | | | |
|---|---|---|---|---|---|---|
| น | า | ย | ก | น | ก | ร |
|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| น | า | ย | ก | น | ก | ร |
|---|---|---|---|---|---|---|

ตัดข้อความเป็นคำ ๆ : Bottom-Up

$$N(k) = \min_{\substack{1 \leq i < k \\ s(i,k) \in \text{Dict}}} \{ N(i-1) + 1 \}, \quad N(0) = 0$$

```
WordSep( dict, s[1..n] ) {  
    N = new array[0..n]  
    N[0] = 0; N[1..n] = ∞  
    I = new array[1..n]  
    for( k = 2; k <= n; k++ )  
        for( i = 1; i < k; i++ )  
            if ( s[i..k] ∈ dict )  
                if ( N[i-1] + 1 < N[k] ) {  
                    N[k] = N[i-1] + 1  
                    I[k] = i  
                }  
    }  
}
```

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 1 | 1 | 4 | 4 | 6 | 4 | I |
| 0 | - | 1 | 1 | 1 | 2 | 2 | 3 | 2 | N |
| | | น | า | ย | ก | น | ก | ร | s |

ตัดข้อความเป็นคำ ๆ : Bottom-Up



```

minWords = an empty list
k = n
while (k > 0) {
    minWords = s[ I[k], k ] + minWords
    k = I[k] - 1
}
return minWords
}

```

Longest Increasing Subsequence

❖ กำหนดให้

❖ A คือลำดับของจำนวน $\langle a_1, a_2, \dots, a_n \rangle$

❖ จงหา

❖ subsequence ที่ยาวสุดของ A ที่มีค่าเรียงจากน้อยไปมาก

❖ ตัวอย่าง

❖ $A = \langle 31, -41, 59, 26, -53, 58, 97, -93, -23, 84 \rangle$

❖ คำตอบคือ $\langle -41, 26, 58, 97 \rangle$

ลองทำดู

Matrix-Chain Multiplication

$$A_1 A_2 A_3 A_4$$

$$(A_1(A_2(A_3A_4)))$$

$$(A_1((A_2A_3)A_4))$$

$$((A_1A_2)(A_3A_4))$$

$$(((A_1A_2)A_3)A_4)$$

$$((A_1(A_2A_3))A_4)$$

วงเล็บกำหนดลำดับการคูณ

ลำดับการคูณต่างกันใช้
เวลาคูณต่างกัน

จงหาวิธีการใส่วงเล็บ
ที่ใช้เวลาการคูณเร็วสุด

Matrix Multiplication

$$\begin{matrix} C & = & A & \times & B \\ [p \times r] & & [p \times q] & & [q \times r] \end{matrix}$$

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;  
}
```

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

ลำดับการคูณต่างกัน ใช้เวลาต่างกัน

$$\begin{matrix} \text{❖ } A_1 & A_2 & A_3 \\ [10 \times 100], & [100 \times 5], & [5 \times 50] \end{matrix}$$

❖ คูณตามลำดับ $((A_1A_2)A_3)$

$$\text{❖ } (A_1A_2) \text{ ต้องคูณด้วย } * \text{ จำนวน } 10 \times 100 \times 5 = 5,000 \text{ ครั้ง}$$

$$\text{❖ ได้เมทริกซ์ } A_{12} \text{ ขนาด } [10 \times 5]$$

$$\text{❖ } (A_{12}A_3) \text{ ต้องคูณด้วย } * \text{ จำนวน } 10 \times 5 \times 50 = 2,500 \text{ ครั้ง}$$

$$\text{❖ รวมเป็น} = \underline{7,500} \text{ ครั้ง}$$

❖ คูณตามลำดับ $(A_1(A_2A_3))$

$$\text{❖ } (A_2A_3) \text{ ต้องคูณด้วย } * \text{ จำนวน } 100 \times 5 \times 50 = 25,000 \text{ ครั้ง}$$

$$\text{❖ ได้เมทริกซ์ } A_{23} \text{ ขนาด } [100 \times 50]$$

$$\text{❖ } (A_1A_{23}) \text{ ต้องคูณด้วย } * \text{ จำนวน } 10 \times 100 \times 50 = 50,000 \text{ ครั้ง}$$

$$\text{❖ รวมเป็น} = \underline{75,000} \text{ ครั้ง}$$

ใส่วงเล็บได้กี่รูปแบบ

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

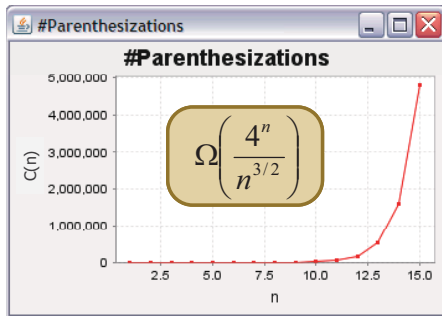
$C(n)$ คือจำนวนการใส่วงเล็บกรณีมี n ตัว

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

$$C(n) = \sum_{k=1}^{n-1} (C(k) C(n-k))$$

ใส่วงเล็บได้กี่รูปแบบ

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



ช้ามาก ๆ ถ้าต้องลองทุกรูปแบบ

นิยาม

- ❖ ต้องการหาวิธีใส่วงเล็บของ $A_1 A_2 \dots A_n$
- ❖ A_1 มีขนาด $p_0 \times p_1$
- ❖ A_2 มีขนาด $p_1 \times p_2$
- ...
- ❖ A_n มีขนาด $p_{n-1} \times p_n$
- ❖ $A_i \dots A_j$ มีขนาด $p_{i-1} \times p_j$
- ❖ ยังไม่หาวิธีการใส่วงเล็บที่ดีที่สุด
- ❖ ขอหาจำนวนการคูณด้วย * ของการใส่วงเล็บที่ดีที่สุด
- ❖ $m(i, j)$ = จำนวน * น้อยสุดเพื่อหาผลคูณ $A_i \dots A_j$
- ❖ $m(1, n)$ คือคำตอบที่ต้องการ

จำนวนการคูณ

$m(i, j)$ คือจำนวนการคูณด้วย * น้อยสุดของการคูณ $A_i A_{i+1} \dots A_j$
 คูณ $A_i \dots A_k$ ใช้ $m(i, k)$ ได้เมทริกซ์ขนาด $[p_{i-1} \times p_k]$
 คูณ $A_{k+1} \dots A_j$ ใช้ $m(k+1, j)$ ได้เมทริกซ์ขนาด $[p_k \times p_j]$
 คูณเมทริกซ์ทั้งสองใช้ $p_{i-1} p_k p_j$

$$\begin{array}{cc} [p_{i-1} \times p_k] & [p_k \times p_j] \\ \underbrace{(A_i \dots A_k)} & \underbrace{(A_{k+1} \dots A_j)} \end{array}$$

$$m(i, k) + m(k+1, j) + p_{i-1} p_k p_j$$

จำนวนการคูณด้วย * น้อยสุด เมื่อแบ่งที่ k

จำนวนการคูณน้อยสุด

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

$$\underbrace{(A_i \dots A_k)}_{m(i, k)} + \underbrace{(A_{k+1} \dots A_j)}_{m(k+1, j)} + p_{i-1} p_k p_j$$

$$(A_1) (A_2 \ A_3 \ A_4 \ A_5) \quad m(1,1) + m(2,5) + p_0 p_1 p_5$$

$$(A_1 \ A_2) (A_3 \ A_4 \ A_5) \quad m(1,2) + m(3,5) + p_0 p_2 p_5$$

$$(A_1 \ A_2 \ A_3) (A_4 \ A_5) \quad m(1,3) + m(4,5) + p_0 p_3 p_5$$

$$(A_1 \ A_2 \ A_3 \ A_4) (A_5) \quad m(1,4) + m(5,5) + p_0 p_4 p_5$$

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

MCM : Top-down

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

```
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;  
}
```

Top-down + Memoization

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

```
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]  
}
```

MCM : Bottom-Up

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

| 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) |

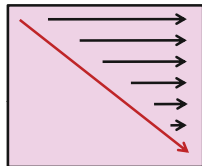
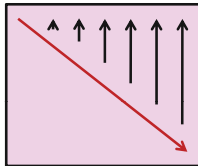
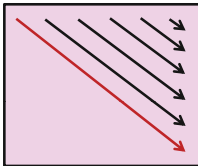
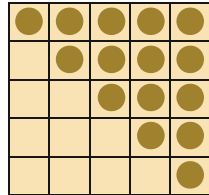
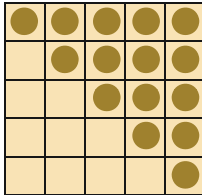
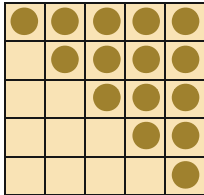
คำตอบใหญ่ได้จากคำตอบย่อย

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

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

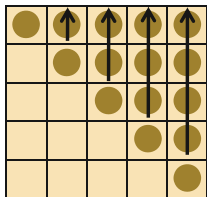
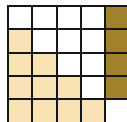
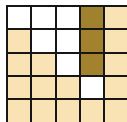
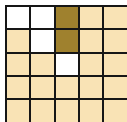
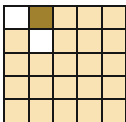
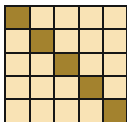
เติมตารางได้หลายรูปแบบ

เติมคำตอบเล็กก่อนคำตอบใหญ่



รูปแบบการเติมตารางคำตอบ

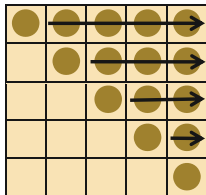
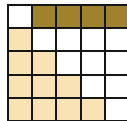
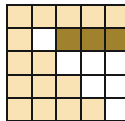
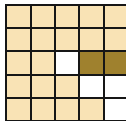
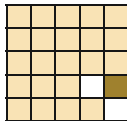
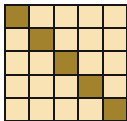
$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$



```
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] = ...
    }
}
```

รูปแบบการเติมตารางคำตอบ

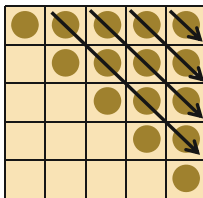
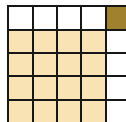
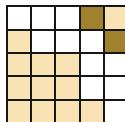
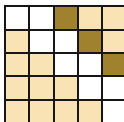
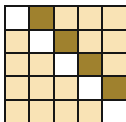
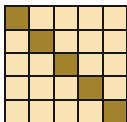
$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$



```
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] = ...
    }
}
```

รูปแบบการเติมตารางคำตอบ

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$



ตัวอย่าง

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

A₁ A₂ A₃ A₄ A₅
10 × 5 × 1 × 5 × 10 × 2

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

ตัวอย่าง

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

A₁ A₂ A₃ A₄ A₅
10 ████ ████ **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 |

$$0 + 25 + 10 \times 5 \times 5 = 275$$

$$50 + 0 + 10 \times 1 \times 5 = 100$$

ตัวอย่าง

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

A_1 A_2 A_3 A_4 A_5
 $10 \times 5 \times \text{[red box]} \times \text{[red box]} \times 10 \times 2$

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

$$0 + 50 + 5 \times 1 \times 10 = 100$$

$$25 + 0 + 5 \times 5 \times 10 = 275$$

ตัวอย่าง

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

A_1 A_2 A_3 A_4 A_5
 $10 \times 5 \times 1 \times \text{ } \times 2$

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

$$0 + 100 + 1 \times 5 \times 2 = 110$$

$$50 + 0 + 1 \times 10 \times 2 = 70$$

ตัวอย่าง

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

A_1 A_2 A_3 A_4 A_5
 $10 \times \text{[grey box]} 10 \times 2$

| j | | 1 | 2 | 3 | 4 | 5 |
|---|--|---|----|-----|-----|-----|
| i | | | | | | |
| 1 | | 0 | 50 | 100 | 200 | |
| 2 | | | 0 | 25 | 100 | |
| 3 | | | | 0 | 50 | 70 |
| 4 | | | | | 0 | 100 |
| 5 | | | | | | 0 |

$$0 + 100 + 10 \times 5 \times 10 = 600$$

$$50 + 50 + 10 \times 1 \times 10 = 200$$

$$100 + 0 + 10 \times 5 \times 10 = 600$$

ตัวอย่าง

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

10 × A₁ 5 × A₂ A₃ A₄ A₅ 2

| 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 |

$$0 + 70 + 5 \times 1 \times 2 = 80$$

$$25 + 100 + 5 \times 5 \times 2 = 175$$

$$100 + 0 + 5 \times 10 \times 2 = 200$$

ตัวอย่าง

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

A_1 A_2 A_3 A_4 A_5
 10 × 2

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

$$0 + 80 + 10 \times 5 \times 2 = 180$$

$$50 + 70 + 10 \times 1 \times 2 = 140$$

$$100 + 100 + 10 \times 5 \times 2 = 300$$

$$200 + 0 + 10 \times 10 \times 2 = 400$$

วงวนการเติมตารางคำตอบ

$$m(i, j) = \begin{cases} \min_{i \leq k \leq j-1} \{ m(i, k) + m(k+1, j) + p_{i-1}p_kp_j \} & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

```
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] = ...
        }
    }
}
```

| 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 | | | |

| | | | | |
|-----|-----|-----|-----|-----|
| 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 |

MCM : Bottom-up

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

```

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;
}

```

 $\Theta(n^3)$

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 |

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

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

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 |

$[1, 5] \rightarrow 2 \quad (A_1 \ A_2) (A_3 \ A_4 \ A_5)$

$[1, 2] \rightarrow 1 \quad ((A_1) (A_2)) (A_3 \ A_4 \ A_5)$

$[3, 5] \rightarrow 4 \quad ((A_1) (A_2)) ((A_3 \ A_4) (A_5))$

$[3, 4] \rightarrow 3 \quad ((A_1) (A_2)) ((A_3) (A_4)) (A_5)$

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;  
}
```

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

MCM : คูณเลข

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

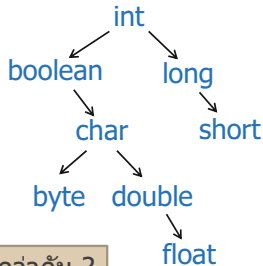
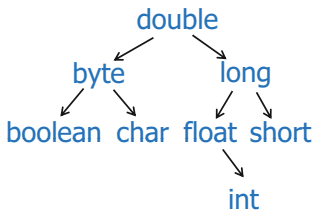
 $\Theta(n^3)$
 $\Theta(?)$

```
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);
}
```

$(A_i \dots A_k)$ $(A_{k+1} \dots A_j)$
 X × Y

Binary Search Tree

- ❖ ต้องการสร้าง **binary search tree** เพื่อเก็บ **keywords** ที่เป็น **primitive data types** ของจาวา
 - ❖ **boolean, byte, char, double, float, int, long, short**



ต้นไหนให้บริการค้นข้อมูลได้ดีกว่ากัน ?

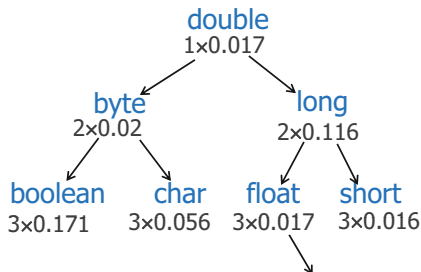
Binary Search Tree

- ❖ เตี้ย \neq ดี เสมอไป : ถ้าพิจารณาความถี่ของค่า
- ❖ ลองนับ **primitive data types** ที่ใช้ใน **source codes** ของ **java.util.* (jdk1.6.0_14)** พบว่า

| | | | | |
|-----------|-------|-------|-------|--------|
| ❖ int | ปรากฏ | 2,274 | ครั้ง | 58.76% |
| ❖ boolean | " | 661 | " | 17.08% |
| ❖ long | " | 450 | " | 11.63% |
| ❖ char | " | 217 | " | 5.61% |
| ❖ byte | " | 78 | " | 2.02% |
| ❖ double | " | 66 | " | 1.71% |
| ❖ float | " | 64 | " | 1.65% |
| ❖ short | " | 60 | " | 1.55% |

Binary Search Tree

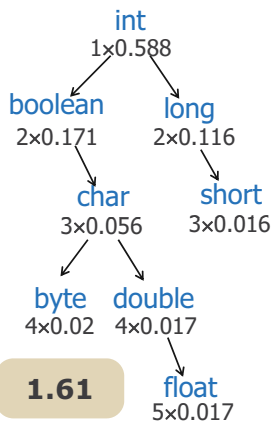
| int | boolean | long | char | byte | double | float | short |
|-------|---------|-------|------|------|--------|-------|-------|
| 58.8% | 17.1% | 11.6% | 5.6% | 2.0% | 1.7% | 1.7% | 1.6% |



$$\sum_{k=1}^n p_k d_k$$

3.42

p_k = ความน่าจะเป็นในการค้นค่าที่ k
 d_k = จำนวนบิตที่อ่านในการค้นค่าที่ k



1.61

Optimal Binary Search Tree

❖ input

- ❖ เขตของค่า และความถี่ในการใช้งานของแต่ละค่า

❖ output

- ❖ binary search tree ที่มี $\sum_{k=1}^n p_k d_k$ น้อยสุด

ลองทำเอง
คล้าย ๆ MCM

