Longest Common Subsequence

Longest Common Subsequence

Subsequence ของ string S คือเซตของอักขระที่ปรากฏในลำดับจากซ้ายไปขวา ไม่จำเป็นต้องมาจากลำดับติดกัน

ตัวอย่างเช่น

ACTTGCC

- ACT, ATTC, T, AC, ACTTGC ทั้งหมดนี้เป็น subsequence
- TTA ไม่ใช่ subsequence

Common subsequence

Common subsequence ของ 2 string คือ subsequence ที่ปรากฏในทั้งสอง string

Longest common subsequence คือ common subsequence ที่มีความยาว(จำนวนตัวอักขระ) มากที่สุด

ตัวอย่าง

S1 = AAACCGTGAGTTATTCGTTCTAGAA

S2 = CACCCCTAAGGTACCTTTGGTTC

ลองหา Longest common subsequence ของ S1 กับ S2

ตัวอย่าง

S1 = AAACCGTGAGTTATTCGTTCTAGAA

S2 = CACCCCTAAGGTACCTTTGGTTC

LCS = ACCTAGTACTTTG

Bruteforce algorithm

สมมติว่า substing x[1...m] และ y[1..n] ที่ต้องการหา LCS

เราจะตรวจสอบทุกๆ subsequence ของ S ว่า เป็น subsequence ของ T หรือไม่

วิเคราะห์

การเปรียบเทียบว่าตรงกันไหมใช้เวลาเท่าไรต่อ subsequence 1 คู่

O(m+n)

พบว่า string ยาว n ตัว สามารถสร้าง substring ได้ 2^n แบบ

ดังนั้น worst case running time = $O((m+n)\ 2^n)$

หา subproblem

พิจารณาความยาวของ longest common subsequence

ใช้ LCS หาค่าของมันเอง

เราจะแทนความยาวของ sequence s ด้วย |s|

แนวทาง เราจะพิจารณา prefixes ของ x และ y

นิยาม c[i,j] = |LCS(x[1..i],y[1..j])|

แล้ว c[m,n] = |LCS(x,y)|

ถ้าให้ $Z=z_1z_2\ldots z_p$ เป็นคำตอบที่ดีที่สุดของ x[1..m] และ y[1..n] แล้ว แบ่งได้ 3 กรณี

1. x[m]=y[n] และ $z_1z_2\ldots z_{p-1}$ จะเป็น LCS ของ x[1..m-1] และ y[1..n-1]
2. x[m]!=y[n] และ $z_1z_2\ldots z_p$ จะเป็น LCS ของ x[1..m-1] และ y[1..n]
3. x[m]!=y[n] และ $z_1z_2\ldots z_p$ จะเป็น LCS ของ x[1..m] และ y[1..n-1]

ถ้าให้ $Z=z_1z_2\ldots z_p$ เป็นคำตอบที่ดีที่สุดของ x[1..m] และ y[1..n] แล้ว แบ่งได้ 3 กรณี

1. x[m]=y[n] และ $z_1z_2\ldots z_{p-1}$ จะเป็น LCS ของ x[1..m-1] และ y[1..n-1]

นั่นคือหาก string x ตัวที่ m และ string y ตัวที่ n เหมือนกัน แสดงว่า คำตอบที่ดีที่สุดคือ 1+ คำตอบที่ดีที่สุดของ x[1..m-1] และ y[1..n-1] หรือ 1+ c[i-1,j-1] นั่นเอง

ถ้าให้ $Z=z_1z_2\ldots z_p$ เป็นคำตอบที่ดีที่สุดของ x[1..m] และ y[1..n] แล้ว แบ่งได้ 3 กรณี

2. x[m]!=y[n] และ $z_1z_2\ldots z_p$ จะเป็น LCS ของ x[1..m-1] และ y[1..n]
3. x[m]!=y[n] และ $z_1z_2\ldots z_p$ จะเป็น LCS ของ x[1..m] และ y[1..n-1]

นั่นคือหาก string x ตัวที่ m และ string y ตัวที่ n **ไม่**เหมือนกัน แสดงว่าคำตอบที่ดีที่สุดคือค่าที่**มากกว่า**ระหว่างคำตอบที่ดีที่สุดของ x[1..m-1] และ y[1..n] กับคำตอบที่ดีที่สุดของ x[1..m] และ y[1..n-1] อย่างใดอย่างหนึ่ง

สังเกตว่า ความยาวของคำตอบที่ดีที่สุด**ไม่**เพิ่มขึ้น (เพราะมันไม่เหมือนกัน)

Recursive formulation

เขียน recurrence ได้ดังนี้

$$c[i,j] = \begin{cases} c[i-1,j-1]+1 & if\, x[i] = y[j] \\ \max\{c[i-1,j], c[i,j-1]\} & \text{กรณีอื่นๆ} \end{cases}$$

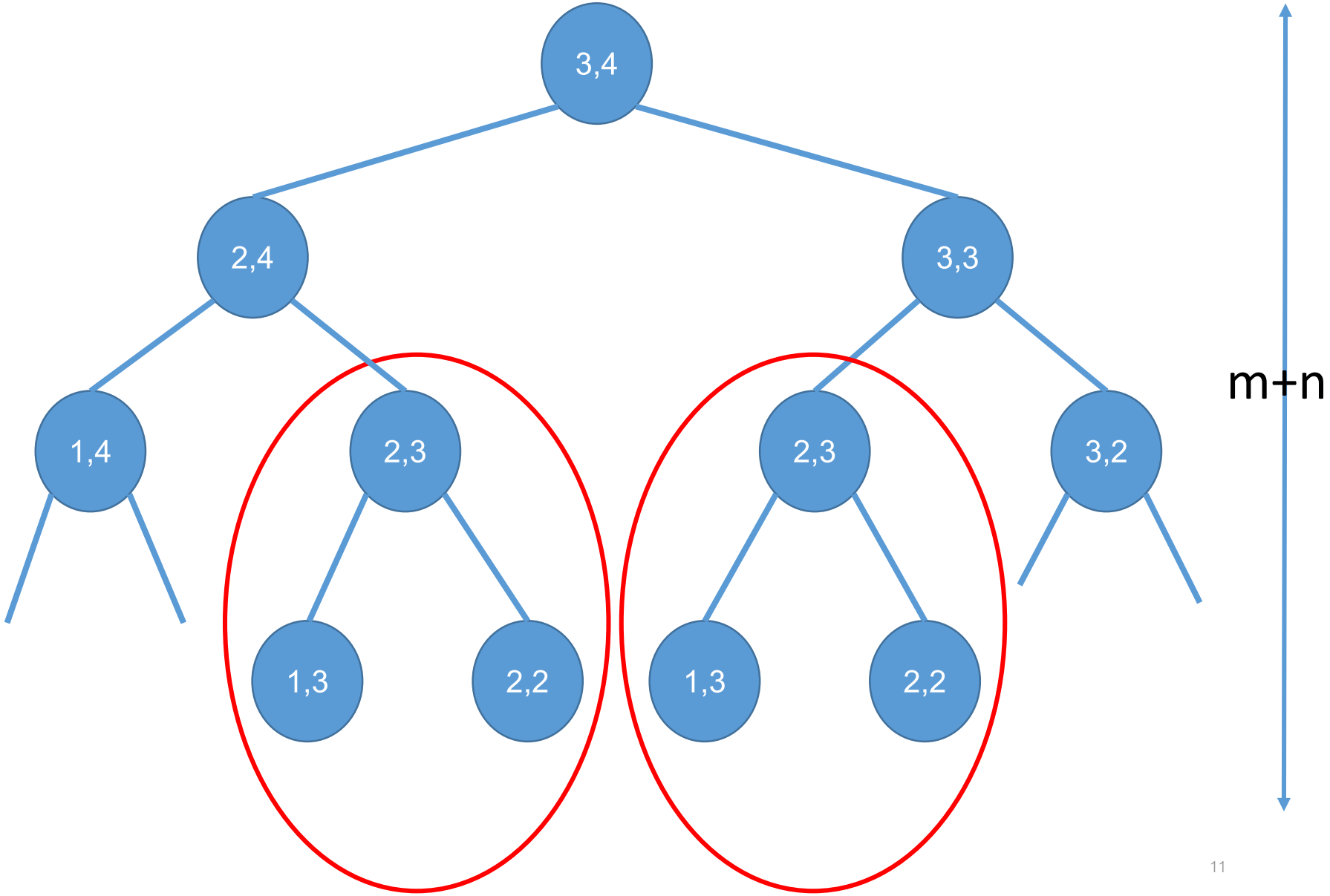
หากเขียน algorithm แบบ recursive จะได้ว่า

```
LCS(x,y,i,j)

if x[i]==y[j] then

        c[i,j] = LCS(x,y,i-1,j-1)+1

else c[i,j] = max{LCS(x,y,i-1,j), LCS(x,y,i,j-1)}
```


3,4
2,4
3,3
1,4
2,3
2,3
3,2
1,3
2,2
1,3
2,2
m+n

Base case

ถ้า string y ไม่มีสักตัว (ยาว=0 ตัว) LCS ควรเป็นเท่าไร

for i=1 to m

    c[i,0] = 0

ถ้า string x ไม่มีสักตัว (ยาว=0 ตัว) LCS ควรเป็นเท่าไร

for j=1 to n

    c[0,j] = 0

```
LCS(x,y)
        for(i=0 to m) c[i,0] = 0
        for(j=0 to n) c[0,j] = 0
        for(i=1 to m)
                for(j=1 to n)
                        if(x[i]=y[j])
                                c[i,j] = c[i-1,j-1]+1
                        else    c[i,j] = max{c[i-1,j], c[i,j-1]}
```

Running Time= O(mn) เนื่องจากคำนวณและเรียกใช้แต่ละช่องเสียเวลาเป็น constant

Space = O(mn)

เริ่มต้น

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
B								
D								
C								
A								
B								
A								

Base case

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**							
D	**0**							
C	**0**							
A	**0**							
B	**0**							
A	**0**							

x[1]!=y[1]

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**	**0**						
D	**0**							
C	**0**							
A	**0**							
B	**0**							
A	**0**							

x[1]=y[2]

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**	**0**	**1**					
D	**0**							
C	**0**							
A	**0**							
B	**0**							
A	**0**							

x[1]!=y[3]

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**	**0**	**1**	**1**				
D	**0**							
C	**0**							
A	**0**							
B	**0**							
A	**0**							

x[1]=y[4]

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**	**0**	**1**	**1**	**1**			
D	**0**							
C	**0**							
A	**0**							
B	**0**							
A	**0**							

x[1]!=y[5]

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**	**0**	**1**	**1**	**1**	**1**		
D	**0**							
C	**0**							
A	**0**							
B	**0**							
A	**0**							

x[1]!=y[6]

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**	**0**	**1**	**1**	**1**	**1**	**1**	
D	**0**							
C	**0**							
A	**0**							
B	**0**							
A	**0**							

x[1]=y[7]

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**
D	**0**							
C	**0**							
A	**0**							
B	**0**							
A	**0**							

เทียบกับ x[2]

- X=BDCABA
- Y=ABCBDAB

		A	B	C	B	D	A	B
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**
D	**0**	**0**	**1**	**1**	**1**	**2**	**2**	**2**
C	**0**							
A	**0**							
B	**0**							
A	**0**							

Fill จนครบ

- X=BDCABA
- Y=ABCBDAB

สร้างตารางย้อนกลับ

เหมือนกัน ให้ใส่ ↖

หากเลือกจากด้านบนให้ใส่ ↑

หากเลือกจากด้านซ้ายให้ใส่ ←

จุดที่เป็น ↖ คือคำตอบ

		A	B	C	B	D	A	B
	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1
D	0	0	1	1	1	2	2	2
C	0	0	1	2	2	2	2	2
A	0	1	1	2	2	2	3	3
B	0	1	2	2	3	3	3	4
A	0	1	2	2	3	3	4	4