12.5. A faster (combinational) algorithm for longest common subsequence

2016101895

Seeun Choi

Index

- What is the longest increasing subsequence
  - Faster construction of the greedy cover
- Longest common subsequence reduces to longest increasing subsequence
- How good is the method
- The lcs of more than two strings

What is the subsequence ?

String1 = mynameisseeun

String2 = yournameissun

In these two strings,

Longest common substring =

What is the subsequence ?

String1 = mynameisseeun

String2 = yournameissun

In these two strings,

Longest common substring = nameiss

What is the subsequence ?

String1 = mynameisseeun
String2 = yournameissun

In these two strings,
Longest common substring = nameiss
Longest common subsequence =

What is the subsequence ?

String1 = mynameisseeun

String2 = yournameissun

In these two strings,

Longest common substring	=	nameiss
Longest common subsequence	=	ynameissun

Index

- **What is the longest increasing subsequence**
    - Faster construction of the greedy cover
- Longest common subsequence reduces to longest increasing subsequence
- How good is the method
- The lcs of more than two strings

Definition

- Let Π be a list of *n* integers, not necessarily distinct.
- An **increasing subsequence** of Π whose values *strictly* increase from left to right.

Π = 5, 3, 4, 9, 6, 2, 1, 8, 7, 10

ex1) {5, 9}
ex2) {5, 9, 10}
ex3) {3, 4, 6, 8, 10}
ex4) {3, 4, 8, 10}
ex5) {6, 8, 10}

Definition

- A **decreasing subsequence** of Π is a subsequence of Π where the numbers are *non-increasing* from left to right.

Π = 4, 8, 3, 9, 5, 2, 5, 3, 10, 1, 9, 1, 6

ex) {4, 2, 1}

{8, 5, 5, 3, 1, 1}

Definition

- A **cover** of Π is a set of decreasing subsequences of that contain all the numbers of Π.

Π = 5, 3, 4, 9, 6, 2, 1, 8, 7, 10

ex) {5, 3, 2, 1} ; {4} ; {9, 6} ; {8, 7} ; {10}

Definition

- the **size** of the cover is the number of decreasing subsequences in it, and a **smallest** cover is a cover with minimum size among all covers.

Π = 5, 3, 4, 9, 6, 2, 1, 8, 7, 10

ex) {5, 3, 2, 1} ; {4} ; {9, 6} ; {8, 7} ; {10}

{5}, {3, 2}, {1}, {4}, {9, 6} , {8}, {7}, {10}

- Naive cover algorithm

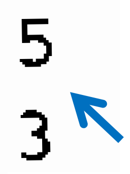
5	4	9	8	10
3		6	7	
2				
1				

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

- Naive cover algorithm

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

- Naive cover algorithm



Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

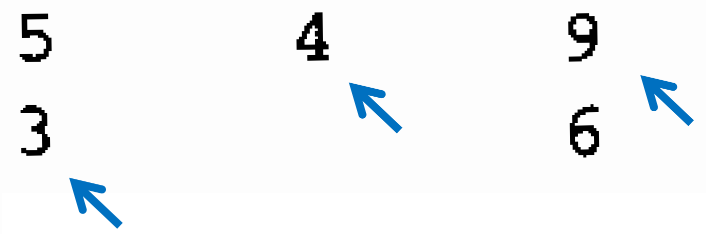
- Naive cover algorithm

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

- Naive cover algorithm

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

- Naive cover algorithm



Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

- Naive cover algorithm

5	4	9
3		6
2		

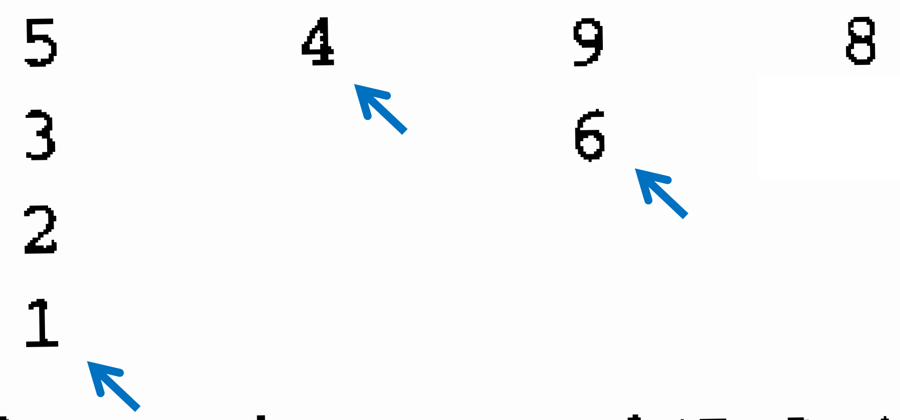
Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

- Naive cover algorithm

5	4	9
3		6
2		
1		

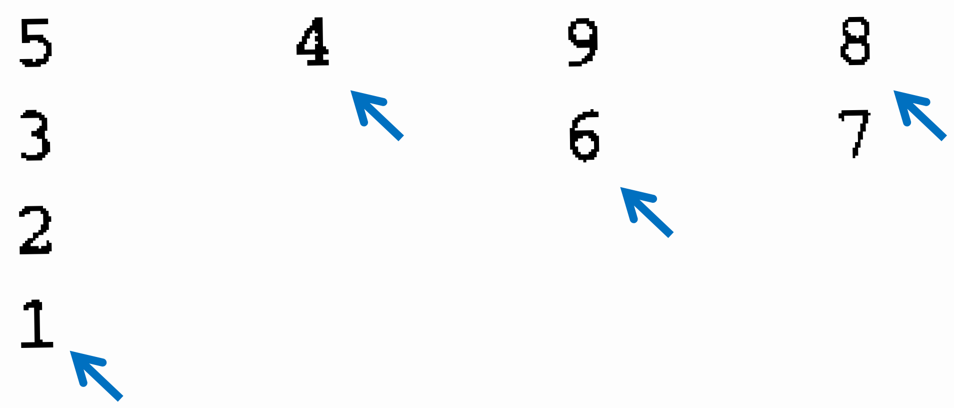
Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

- Naive cover algorithm



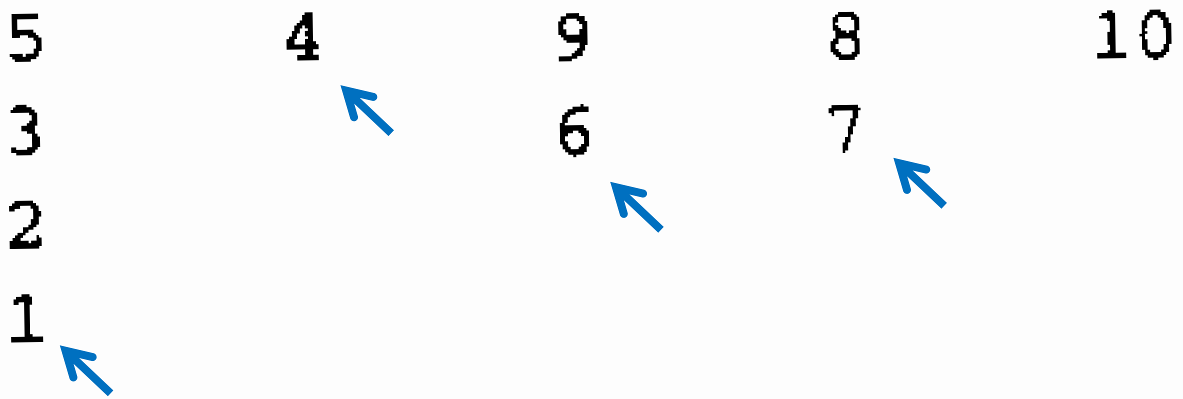
Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

- Naive cover algorithm



Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

- Naive cover algorithm



Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

Lemma 12.5.1.

- If I is an increasing subsequence of Π with length equal to the size of a cover of Π, call it C, then I is a longest increasing subsequence of Π and C is a smallest cover of Π.

proof

- **What is the length of I and size of cover ?**
- How to prove the premise ?
- How to prove the lemma 12.5.1 ?

$\Pi = 5, 3, 4, 9, 6, 2, 1, 8, 7, 10$

length of I	ex1)	{5, 9}	length of I	=	2
	ex2)	{5, 9, 10}	length of I	=	3
	ex3)	{3, 4, 6, 8, 10}	length of I	=	5
	ex4)	{3, 4, 8, 10}	length of I	=	4
	ex5)	{6, 8, 10}	length of I	=	3

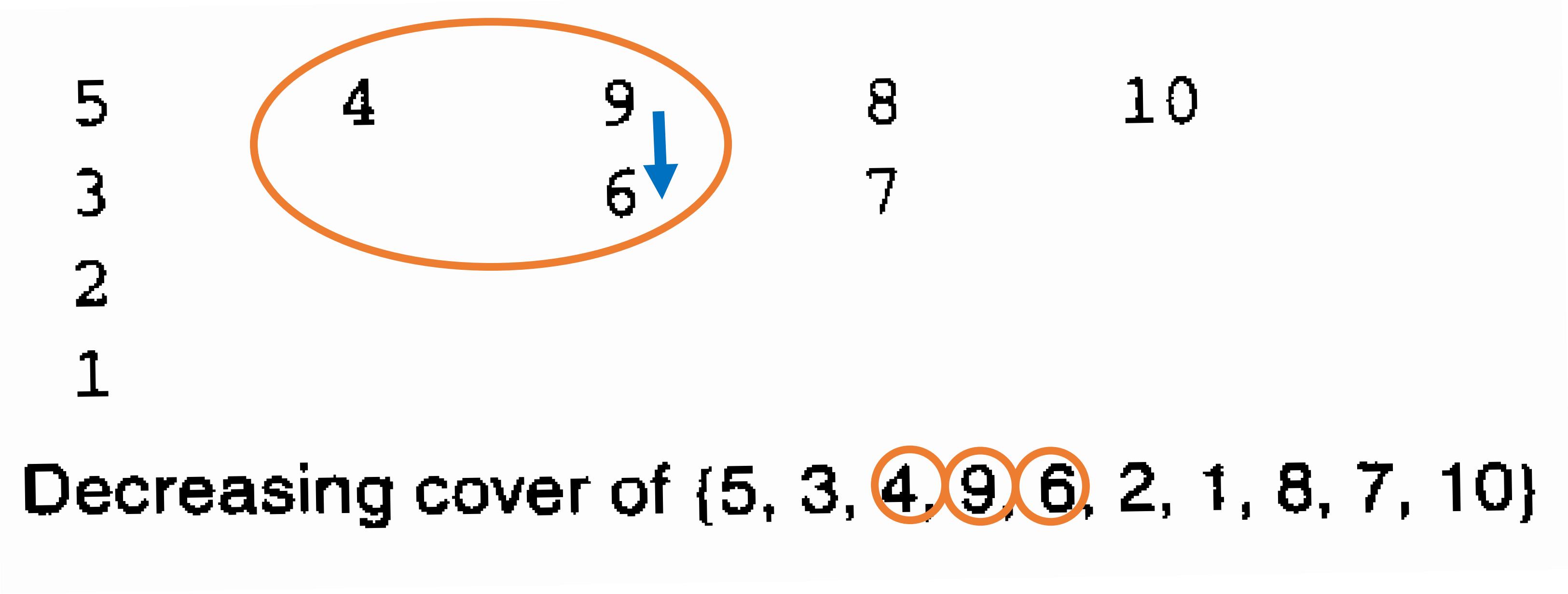
size of a cover of Π	ex1) {5, 3, 2, 1} ; {4} ; {9, 6} ; {8, 7} ; {10}	5
	ex2) {5}, {3, 2}, {1}, {4}, {9, 6} , {8}, {7}, {10}	8

Lemma 12.5.1.

- If I is an increasing subsequence of Π with length equal to the size of a cover of Π, call it C, then I is a longest increasing subsequence of Π and C is a smallest cover of Π.

proof

- What is the length of I and size of cover ?
- **How to prove the premise ?**
- How to prove the lemma 12.5.1 ?

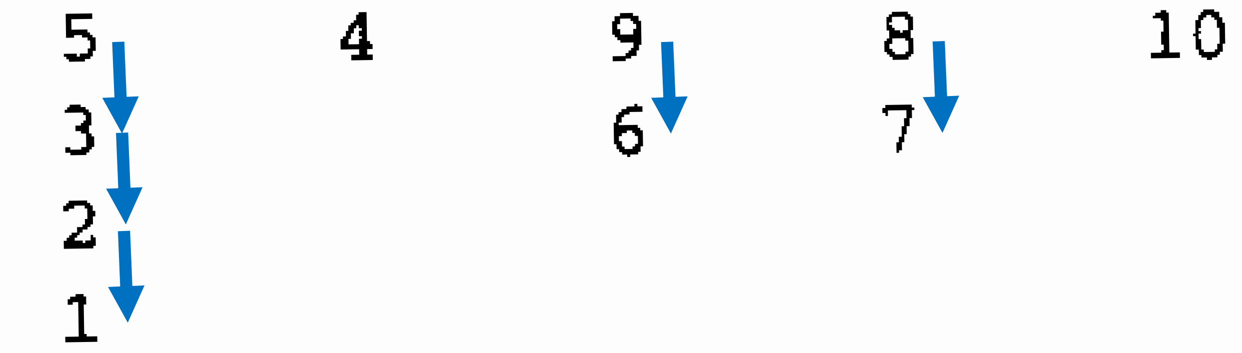


premise) Increasing subsequence I 는 decreasing subsequence 에 포함된 수를 하나 보다 더 포함할 수 없다.

proof) I 가 decreasing subsequence 의 집합인 cover 의
각 decreasing subsequence 에서 두 개의 수를 포함할 경우

I = 4, 9, 6

error -> I is not increasing



Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

premise) Increasing subsequence I 는 decreasing subsequence 에 포함된 수를 하나 보다 더 포함할 수 없다.

∴ I.length() ≤ C.size()

- A **cover** of Π is a set of decreasing subsequences of that contain all the numbers of Π.

Lemma 12.5.1.

- If I is an increasing subsequence of Π with length equal to the size of a cover of Π, call it C, then I is a longest increasing subsequence of Π and C is a smallest cover of Π.

proof

- What is the length of I and size of cover ?
- How to prove the premise ?
- **How to prove the lemma 12.5.1 ?**

assumption) I 의 길이가 cover 의 크기와 같다

I 는 longest increasing subsequence 이다.

∵ I.length() ≤ C.size() in the premise

conversely, $C' < C$ 인 C' 가 존재한다.

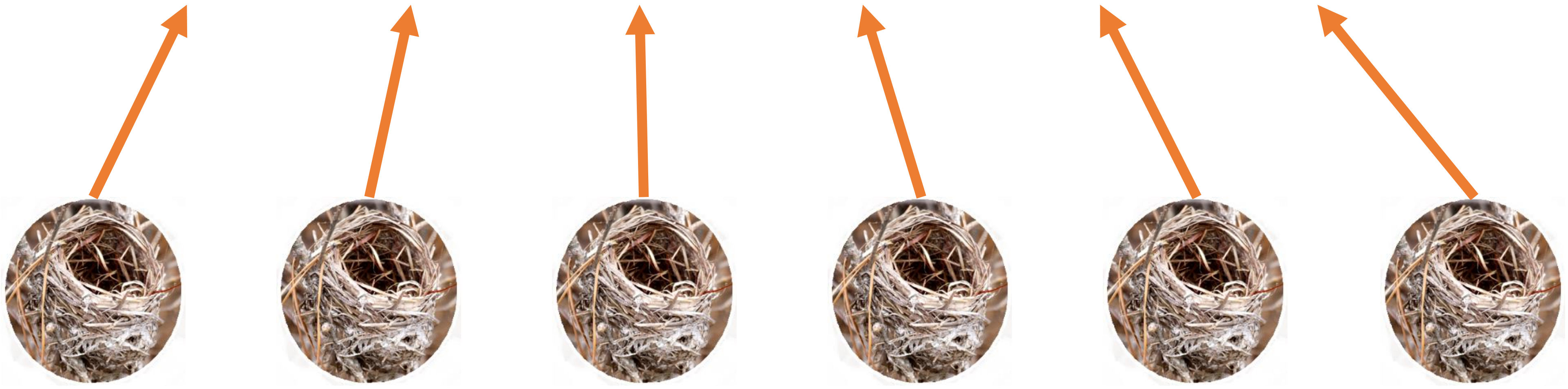
5	4	9	2	8	10
3		6	1	7	

size of cover = 6

length of I = 6

D Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

The Pigeonhole Principle



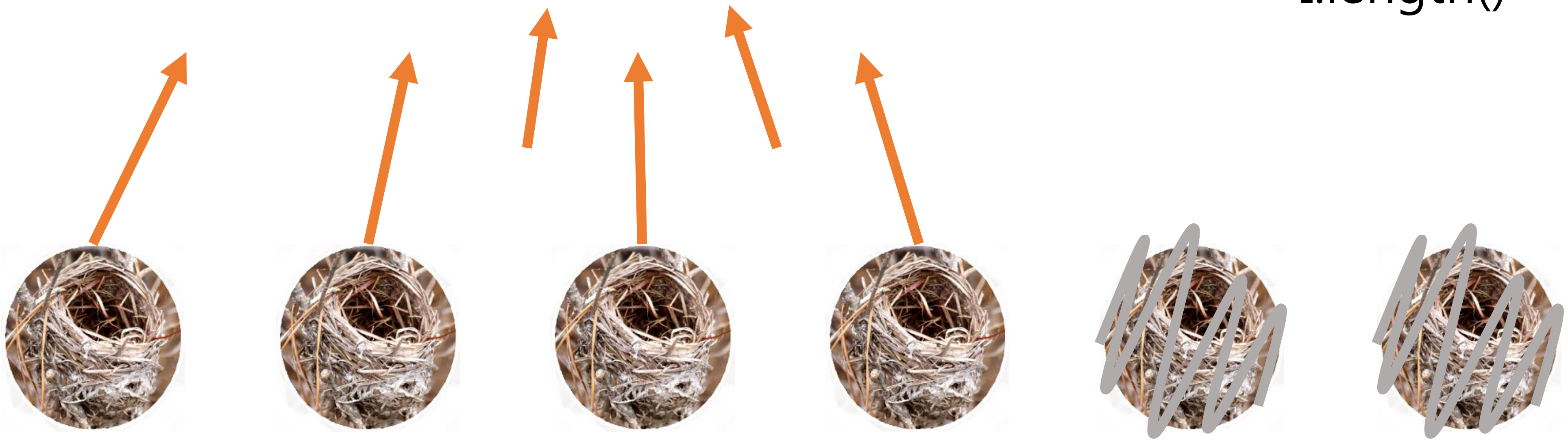
5	4	9	2	8	10
3		6	1	7	

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

size of cover = 6
length of I = 6

C′ = 4

I.length() > C′.size()
-> error



assumption)　I 의 길이가 cover 의 크기와 같다

I 는 longest increasing subsequence 이다.

$\because$ I.length() $\leq$ C.size() in the premise

conversely,　C′ < C 인 C′ 가 존재한다.

-> error

$\therefore$ I 의 길이가 cover 의 크기와 같으면 I 는 longest subsequence 이고 C 는 smallest cover 이다.

Lemma 12.5.2.

- The greedy cover of Π can be built in O(n^2) time.

ex) Π = 1, 2, 3, 4

1				0
1	2			1
1	2	3		2
1	2	3	4	3

$$0+1+2+\ldots+(n-1) = (n-1)n/2$$

Lemma 12.5.3.

- There is an increasing subsequence I of Π containing exactly one number from each decreasing subsequence in the greedy cover C. Hence I is the longest possible, and C is the smallest possible.

LIS algo.

Longest increasing subsequence algorithm

begin

0. Set i to be the number of subsequences in the greedy cover. Set I to the empty list; pick any number x in subsequence i and place it on the front of list I.

1. While $i > 1$ do
begin

2. Scanning down from the *top* of subsequence $i - 1$, find the first number y that is smaller than x.

3. Set x to y and i to $i - 1$.

4. Place x on the front of list I.
end

end.

5	4	9	8	10
3		6	7	
2				
1				

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

I = { }

5	4	9	8	10
3		6	7	
2				
1				

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

I = {10}

5	4	9	8	10
3		6	7	
2				
1				

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

I = {8, 10}

5	4	9	8	10
3		6	7	
2				
1				

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

I = {8, 10}

5	4	9	8	10
3		6	7	
2				
1				

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

$I = \{6, 8, 10\}$

5	4	9	8	10
3		6	7	
2				
1				

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

$I = \{4, 6, 8, 10\}$

5	4	9	8	10
3		6	7	
2				
1				

Decreasing cover of $\{5, 3, 4, 9, 6, 2, 1, 8, 7, 10\}$

$I = \{4, 6, 8, 10\}$

5	4	9	8	10
3		6	7	
2				
1				

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

$I = \{3, 4, 6, 8, 10\}$

Index

- What is the longest increasing subsequence
  - **Faster construction of the greedy cover**
- Longest common subsequence reduces to longest increasing subsequence
- How good is the method
- The lcs of more than two strings

Theorem 12.5.1.

- The greedy cover can be constructed in O($n \log n$) time. A longest increasing subsequence and a smallest cover of Π can therefore be found in O($n \log n$)

proof

Definition

- L is the ordered list containing the last number of each of the decreasing subsequences built so far.

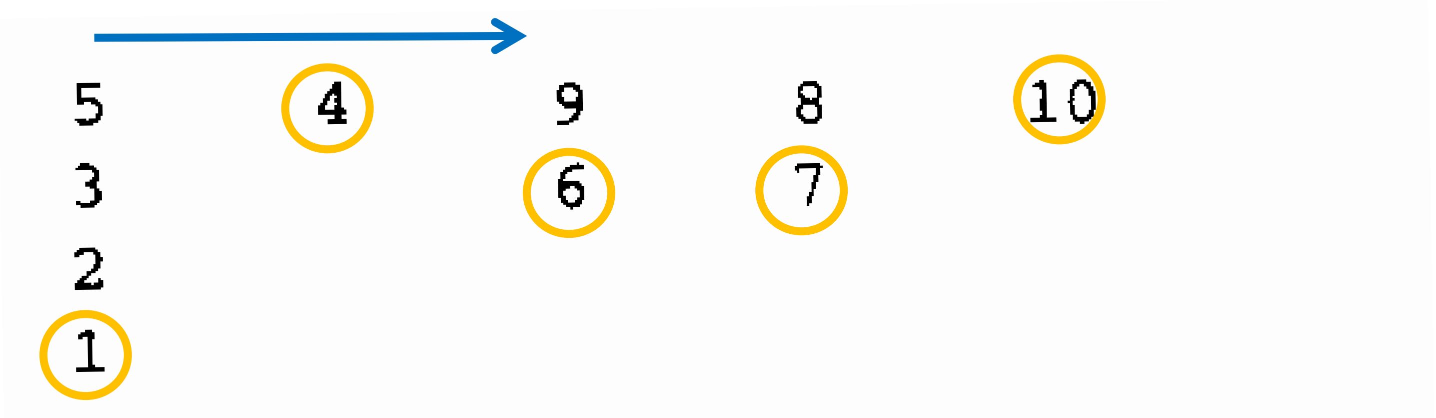
5	4	9	8	10
3		6	7	
2				
1				

L = {1, 4, 6, 7, 10}

Lemma 12.5.4.

- At any point in the execution of the algorithm, the list L is sorted in increasing order.

proof



L = {1, 4, 6, 7, 10}

- Naive cover algorithm

ex1

5	4	9
3		6

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

L = {3, 4, 6}

binary search

- Naive cover algorithm

ex1

5	4	9
3		6
2		

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

L = {2, 4, 6}

- Naive cover algorithm

ex2

5	4	9	8
3		6	
2			
1			

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

L = {1, 4, 6, 8}

binary search

- Naive cover algorithm

$p \rightarrow \log p$

ex2

5	4	9	8
3		6	7
2			
1			

Decreasing cover of {5, 3, 4, 9, 6, 2, 1, 8, 7, 10}

$L = \{\underbrace{1,\ 4,\ 6,\ 7}_{p}\}$

binary search

Theorem 12.5.1.

- The greedy cover can be constructed in $\mathrm{O}(\mathrm{n}\log n)$ time. A longest increasing subsequence and a smallest cover of Π can therefore be found in $\mathrm{O}(\mathrm{n}\log n)$

- In fact, if p is the length of the LIS, then it can be found in $\mathrm{O}(\mathrm{n}\log p)$ time.

$$\mathrm{O}(\mathrm{n}\log p) \xrightarrow{p \leq n} \mathrm{O}(\mathrm{n}\log n)$$

Index

- What is the longest increasing subsequence
  - Faster construction of the greedy cover
- **Longest common subsequence reduces to longest increasing subsequence**
- How good is the method
- The lcs of more than two strings

Theorem 12.5.2.

- Every increasing subsequence I in $\Pi(S_1, S_2)$ specifies an equal length common subsequence of S_1 and S_2 and vice versa. Thus a longest common subsequence of S_1 and S_2 corresponds to a longest increasing subsequence in the list $\Pi(S_1, S_2)$.

proof

- **What is $\boldsymbol{\Pi(S_1, S_2)}$?**
- Correlation between CS and IS
- How to solve LCS using LIS ?

Definition

- Given strings S_1 and S_2 (of length m and n, respectively) over at alphabet Σ, let $\mathrm{r(i)}$ be the number of times that the i th character of string S_1 appears in string S_2.

S_1 = abacx

S_2 = baabca

$$\mathrm{r}(1) = 3, \mathrm{r}(2) = 2, \mathrm{r}(3) = 3, \mathrm{r}(4) = 1, \mathrm{r}(5) = 0$$

Definition

- Given strings S_1 and S_2 (of length m and n, respectively) over at alphabet $\sum$, let $\mathrm{r(i)}$ be the number of times that the i th character of string S_1 appears in string S_2.

S_1 = abacx

S_2 = baabca

$\mathrm{r}(1) = 3$

Definition

- Given strings S_1 and S_2 (of length m and n, respectively) over at alphabet $\sum$, let $\mathrm{r(i)}$ be the number of times that the i th character of string S_1 appears in string S_2.

S_1 = abacx

S_2 = baabca

$\mathrm{r}(1) = 3, \mathrm{r}(2) = 2$

Definition

- Given strings S_1 and S_2 (of length m and n, respectively) over at alphabet $\sum$, let $\mathrm{r(i)}$ be the number of times that the i th character of string S_1 appears in string S_2.

S_1 = abacx

S_2 = baabca

$\mathrm{r(1) = 3, r(2) = 2, r(3) = 3}$

Definition

- Given strings S_1 and S_2 (of length m and n, respectively) over at alphabet $\sum$, let $\mathrm{r(i)}$ be the number of times that the i th character of string S_1 appears in string S_2.

S_1 = abacx

S_2 = baabca

$$\mathrm{r(1) = 3, r(2) = 2, r(3) = 3, r(4) = 1}$$

Definition

- Given strings S_1 and S_2 (of length m and n, respectively) over at alphabet $\sum$, let $\mathrm{r(i)}$ be the number of times that the i th character of string S_1 appears in string S_2.

S_1 = abacx

S_2 = baabca

$$\mathrm{r(1) = 3, r(2) = 2, r(3) = 3, r(4) = 1, r(5) = 0}$$

Definition

- Let r denote the sum $\sum_{i=1}^{m} r(i)$.

S_1 = abacx

S_2 = baabca

$\mathrm{r}(1) = 3, \mathrm{r}(2) = 2, \mathrm{r}(3) = 3, \mathrm{r}(4) = 1, \mathrm{r}(5) = 0$

$\mathrm{r} = 3{+}2{+}3{+}1{+}0 = 9$

Definition

- $\Pi(S_1, S_2)$ is a list with length r, in which each character instance in S_1 is replaced with the associated list for that character, in decreasing order.
  That is, for each position i in S_1, insert the list associated with the character $S_1(i)$.

S_1 = abacx

S_2 = baabca

$S_1(1)$ = a

$\Pi(S_1, S_2)$ = 6, 3, 2, 4, 1, 6, 3, 2, 5

Definition

- $\Pi(S_1, S_2)$ is a list with length r, in which each character instance in S_1 is replaced with the associated list for that character, in decreasing order.
  That is, for each position i in S_1, insert the list associated with the character $S_1(i)$.

S_1 = abacx

S_2 = baabca

$S_1(1)$ = a $S_1(2)$ = b

$\Pi(S_1, S_2)$ = 6, 3, 2, 4, 1, 6, 3, 2, 5

Definition

- $\Pi(S_1, S_2)$ is a list with length r, in which each character instance in S_1 is replaced with the associated list for that character, in decreasing order.
  That is, for each position i in S_1, insert the list associated with the character $S_1(i)$.

S_1 = abacx

S_2 = baabca

$S_1(1)$ = a $S_1(2)$ = b $S_1(3)$ = a

$\Pi(S_1, S_2)$ = 6, 3, 2, 4, 1, 6, 3, 2, 5

Definition

- $\Pi(S_1, S_2)$ is a list with length r, in which each character instance in S_1 is replaced with the associated list for that character, in decreasing order.
  That is, for each position i in S_1, insert the list associated with the character $S_1(i)$.

S_1 = abacx

S_2 = baabca

$S_1(1)$ = a $S_1(2)$ = b $S_1(3)$ = a $S_1(4)$ = c

$\Pi(S_1, S_2)$ = 6, 3, 2, 4, 1, 6, 3, 2, 5

Definition

- $\Pi(S_1, S_2)$ is a list with length r, in which each character instance in S_1 is replaced with the associated list for that character, in decreasing order.
  That is, for each position i in S_1, insert the list associated with the character $S_1(i)$.

S_1 = abacx

S_2 = baabca

$S_1(1)$ = a $S_1(2)$ = b $S_1(3)$ = a $S_1(4)$ = c $S_1(5)$ = x

$\Pi(S_1, S_2)$ = 6, 3, 2, 4, 1, 6, 3, 2, 5

Definition

- $\Pi(S_1, S_2)$ is a list with length r, in which each character instance in S_1 is replaced with the associated list for that character, in decreasing order.
  That is, for each position i in S_1, insert the list associated with the character $S_1(i)$.

S_1 = abacx

S_2 = baabca

$S_1(1)$ = a $S_1(2)$ = b **r** $S_1(3)$ = a $S_1(4)$ = c $S_1(5)$ = x

$\Pi(S_1, S_2) = \underbrace{6, 3, 2, 4, 1, 6, 3, 2, 5}$

Theorem 12.5.2.

- Every increasing subsequence I in $\Pi(S_1, S_2)$ specifies an equal length common subsequence of S_1 and S_2 and vice versa. Thus a longest common subsequence of S_1 and S_2 corresponds to a longest increasing subsequence in the list $\Pi(S_1, S_2)$.

proof

- What is $\Pi(S_1, S_2)$?
- **Correlation between CS and IS**
- How to solve LCS using LIS ?

S_1 = abacx

S_2 = baabca

$$\Pi(S_1, S_2) = \overset{a}{6, 3, 2,} \,/\, \overset{b}{4, 1,} \,/\, \overset{a}{6, 3, 2,} \,/\, \overset{c}{5}$$

S_1 = abacx

S_2 = baabca

CS = bac

a b a c

$\Pi(S_1, S_2) = 6, 3, 2, 4, 1, 6, 3, 2, 5$

S_1 = abacx

S_2 = baabca

CS = bac

a b a c

$$\Pi(S_1, S_2) = 6, 3, 2, 4, 1, 6, 3, 2, 5$$

S_1 = abacx

S_2 = baabca

CS = bac

$$\Pi(S_1, S_2) = \overset{a}{6,\ 3,\ 2},\ \big/\ \overset{b}{4,\ 1},\ \big/\ \overset{a}{6,\ 3,\ 2},\ \big/\ \overset{c}{5}$$

S_1 = abacx

S_2 = baabca

CS = bac

a b a c

$\Pi(S_1, S_2)$ = 6, 3, 2, / 4, 1, / 6, 3, 2, / 5

S_1 = abacx

S_2 = baabca

CS = bac

a b a c

$\Pi(S_1, S_2)$ = 6, 3, 2, / 4, 1, / 6, 3, 2, / 5

S_1 = abacx

S_2 = baabca

CS = bac

$$\begin{array}{l} \qquad\quad \text{a} \qquad\qquad \text{b} \qquad\qquad \text{a} \qquad\qquad \text{c} \\ \Pi(S_1, S_2) = 6, 3, 2, / 4, 1, / 6, 3, 2, / 5 \end{array}$$

1, 2, 5 = IS

a b a c

$\Pi(S_1, S_2)$ = 6, 3, 2 / 4, 1 / 6, 3, 2 / 5

1, 2, 5 = IS

S_1 = abacx

S_2 = baabca

a b a c

$\Pi(S_1, S_2)$ = 6, 3, 2, 4, 1, 6, 3, 2, 5

1, 2, 5 = IS

S_1 = abacx

S_2 = baabca

a b a c

$\Pi(S_1, S_2) = 6, 3, 2, \,/\, 4, 1, \,/\, 6, 3, 2, \,/\, 5$ 1, 2, 5 = IS

S_1 = abacx

S_2 = baabca

a b a c

$\Pi(S_1, S_2) = 6, 3, 2, / 4, 1, / 6, 3, 2, / 5$

1, 2, 5 = IS

S_1 = abacx

S_2 = baabca

CS = bac

∴ We can solve LCS using LIS

Theorem 12.5.2.

- Every increasing subsequence I in $\Pi(S_1, S_2)$ specifies an equal length common subsequence of S_1 and S_2 and vice versa. Thus a longest common subsequence of S_1 and S_2 corresponds to a longest increasing subsequence in the list $\Pi(S_1, S_2)$.

proof

- What is $\Pi(S_1, S_2)$?
- Correlation between CS and IS
- **How to solve LCS using LIS ?**

S_1 = abacx

S_2 = baabca

$\Pi(S_1, S_2)$ = 6, 3, 2, / 4, 1, / 6, 3, 2, / 5

cover

6	4	6
3	3	5
2	2	
1		

Index1 = 1

Index2 = 3

S = a

I = {3, 4, 5}

S_1 = abacx

S_2 = baabca

$\Pi(S_1, S_2)$ = 6, 3, 2, 4, 1, 6, 3, 2, 5

cover

6	4	6
3	3	5
2	2	
1		

I = {3, 4, 5}

Index1 = 1 , 2
Index2 = 3 , 4
S = a , b

S_1 = abacx

S_2 = baabca

$\Pi(S_1, S_2) = 6, 3, 2, \,/\, 4, 1, \,/\, 6, 3, 2, \,/\, 5$

cover	6	4	6
	3	3	5
	2	2	
	1		

I = {3, 4, 5}

Index1 = 1 , 2 , 4
Index2 = 3 , 4 , 5
S = a , b , c

∴ LCS = abc

Theorem 12.5.3.

- The longest common subsequence problem can be solved in O(r $\log n$) time.

In greedy cover O(n $\log n$)

Π 의 길이

LIS 생성 (binary search)

$$\Pi(S_1, S_2) = \{\overbrace{\qquad\qquad\qquad}^{r}\}$$

O(r $\log l$)

$l \le n$

O(r $\log n$)

LCS = LIS

Π 의 길이

Index

- What is the longest increasing subsequence
  - Faster construction of the greedy cover
- Longest common subsequence reduces to longest increasing subsequence
- **How good is the method**
- The lcs of more than two strings

Time complexity of LCS is $O(r \log n)$

r is expected to be nm/σ

$$\therefore O(\frac{nm}{\sigma} \log n)$$

$O(\frac{nm}{\sigma} \log n)$ looks attractive compared to $O(nm)$

Traceback example

	Ø	A	G	C	A	T
Ø	0	0	0	0	0	0
G	0	↑←0	↖1	←1	←1	←1
A	0	↖1	↑←1	↑←1	↖2	←2
C	0	↑1	↑←1	↖2	↑←2	↑←2

Dynamic programming -> O(nm)

$S_1 = \square\ \square\ \square\ \square\ \square$ m

$\Big\downarrow \frac{1}{\sigma}$

$S_2 = \square\ \square\ \square\ \square\ \square\ \square$ n

S_1 = □ □ □ □ □ m

$\Big\downarrow \frac{1}{\sigma}$ $\sigma \fallingdotseq 100$

S_2 = □ □ □ □ □ □ n

σ is what roman alphabet with capital letters, digits, and punctuation marks added

$S_1 = \square\ \square\ \square\ \square\ \square$ m

$$S_1 \xrightarrow{\frac{1}{\sigma}} S_2 \qquad \sigma \fallingdotseq 100 \qquad r = \frac{nm}{\sigma} = \frac{nm}{100}$$

$S_2 = \square\ \square\ \square\ \square\ \square\ \square$ n

$$\mathrm{r}\log n = \frac{nm}{100}\log n < nm$$

$\therefore$ $O(\frac{nm}{\sigma}\log n)$ looks attractive compared to O(nm)

Index

- What is the longest increasing subsequence
  - Faster construction of the greedy cover
- Longest common subsequence reduces to longest increasing subsequence
- How good is the method
- **The lcs of more than two strings**

Theorem 12.5.4.

- Every increasing subsequence in $\Pi(S_1, S_2, S_3)$ specifies an equal length common subsequence of S_1, S_2, S_3 and vice versa. Therefore, a longest common subsequence of S_1, S_2, S_3 corresponds to a longest increasing subsequence in $\Pi(S_1, S_2, S_3)$.

S_1 = abacx

S_2 = baabca

S_3 = babbac

S_1 = abacx

S_2 = baabca   6, 3, 2

S_3 = babbac   5, 2

$$\Pi(S_1, S_2, S_3) = (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2)$$

S_1 = abacx

S_2 = baabca   4, 1

S_3 = babbac   4, 3, 1

$\Pi(S_1, S_2, S_3) =$ (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2),

(4, 4), (4, 3), (4, 1), (1, 4), (1, 3), (1, 1),

S_1 = abacx

S_2 = baabca   6, 3, 2

S_3 = babbac   5, 2

$\Pi(S_1, S_2, S_3) =$ (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2)$\cancel{,}$

(4, 4), (4, 3), (4, 1), (1, 4), (1, 3), (1, 1)$\cancel{,}$

(6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2)

S_1 = abacx

S_2 = baabca 5

S_3 = babbac 6

$$\Pi(S_1, S_2, S_3) = (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2),$$
$$(4, 4), (4, 3), (4, 1), (1, 4), (1, 3), (1, 1),$$
$$(6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2),$$
$$(5, 6)$$

S_1 = abacx

S_2 = baabca

S_3 = babbac

$$\Pi(S_1, S_2, S_3) = (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2)\cancel{,}$$

$$(4, 4), (4, 3), (4, 1), (1, 4), (1, 3), (1, 1)\cancel{,}$$

$$(6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2)\cancel{,}$$

$$(5, 6)$$

S_1 = abacx

S_2 = baabca

S_3 = babbac

$\Pi(S_1, S_2, S_3) =$ (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2),

(4, 4), (4, 3), (4, 1), (1, 4), (1, 3), (1, 1), **b**

(6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2), **a**

(5, 6) **c**

S_1 = abacx

S_2 = baabca

S_3 = babbac

$\Pi(S_1, S_2, S_3) =$ (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2),

(4, 4), (4, 3), (4, 1), $\underline{(1, 4)}$, (1, 3), (1, 1),

(6, 5), (6, 2), $\underline{(3, 5)}$, (3, 2), (2, 5), (2, 2),

$\underline{(5, 6)}$

b

a

c

S_1 = abacx

S_2 = baabca

S_3 = babbac

$$\Pi(S_1, S_2, S_3) = (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2),$$

$$(4, 4), (4, 3), (4, 1), \underline{(1, 4)}, (1, 3), (1, 1),$$

$$(6, 5), (6, 2), \underline{(3, 5)}, (3, 2), (2, 5), (2, 2),$$

$$\underline{(5, 6)}$$

b

a

c

S_1 = abacx

S_2 = baabca

S_3 = babbac

$\Pi(S_1, S_2, S_3)$ = (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2),

(4, 4), (4, 3), (4, 1), (1, 4), (1, 3), (1, 1), **b**

(6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2), **a**

(5, 6) **c**

$$S_1 = \text{abacx}$$

$$S_2 = \text{baabca}$$

$$S_3 = \text{babbac}$$

$$\Pi(S_1, S_2, S_3) = (6, 5), (6, 2), (3, 5), (3, 2), (2, 5), (2, 2)\cancel{,}$$

$$(4, 4), (4, 3), (4, 1), \underline{(1, 4)}, (1, 3), (1, 1)\cancel{,} \qquad \mathbf{b}$$

$$(6, 5), (6, 2), \underline{(3, 5)}, (3, 2), (2, 5), (2, 2)\cancel{,} \qquad \mathbf{a}$$

$$\underline{(5, 6)} \qquad \mathbf{c}$$