DYNAMIC PROGRAMMING: EVEN MORE FUN!

David Kauchak
CS 140 – Spring 2023

Admin

Assignment 7

Assignment 8

Longest increasing subsequence

Given a sequence of numbers $X = x_1, x_2, \ldots, x_n$ find the longest increasing *subsequence* $(i_1, i_2, \ldots, i_m)$, that is a subsequence where numbers in the sequence increase.

5 2 8 6 3 6 9 7

Longest increasing subsequence

Given a sequence of numbers $X = x_1, x_2, \ldots, x_n$ find the longest increasing *subsequence* $(i_1, i_2, \ldots, i_m)$, that is a subsequence where numbers in the sequence increase.

5 2 8 6 3 6 9 7

1b: recursive solution

5 2 8 6 3 6 9 7

↑

Is 5 part off the LIS?

1b: recursive solution

5 2 8 6 3 6 9 7

↑

Two options:
Either 5 is in the
LIS or it's not

1b: recursive solution

5 2 8 6 3 6 9 7

include 5 ↑

5 + LIS(8 6 3 6 9 7)

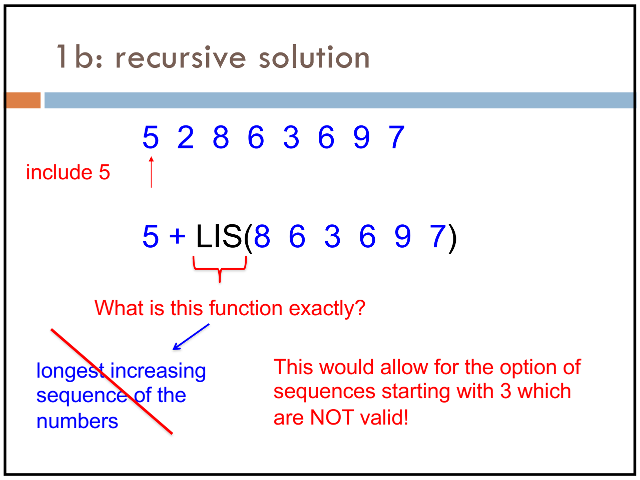
1b: recursive solution

5 2 8 6 3 6 9 7

include 5 ↑

$$5 + \underbrace{\text{LIS}}_{\text{What is this function exactly?}}(8\ \ 6\ \ 3\ \ 6\ \ 9\ \ 7)$$

longest increasing sequence of the numbers	longest increasing sequence of the numbers starting with 8



1b: recursive solution

5 2 8 6 3 6 9 7

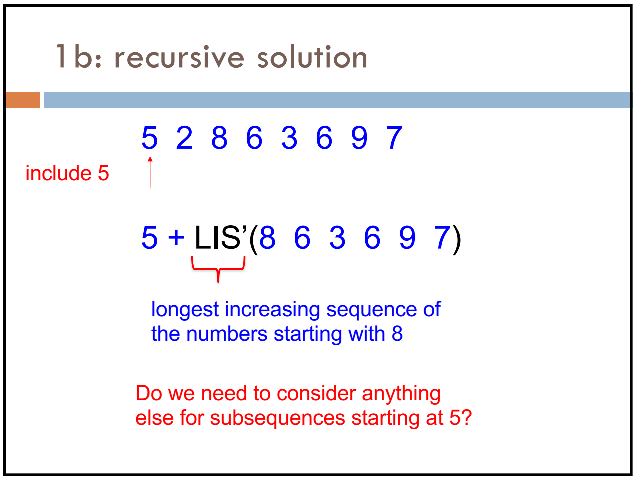
include 5 ↑

$$5 + \underbrace{\text{LIS}}(8\ \ 6\ \ 3\ \ 6\ \ 9\ \ 7)$$

What is this function exactly?

~~longest increasing sequence of the numbers~~

This would allow for the option of sequences starting with 3 which are NOT valid!



1b: recursive solution

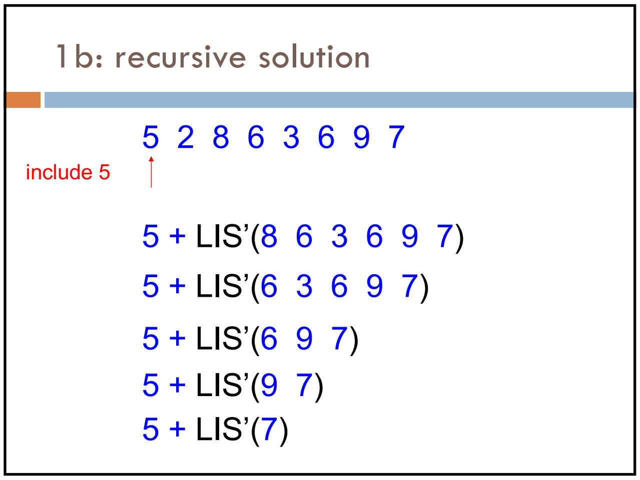
5 2 8 6 3 6 9 7

include 5 ↑

$$5 + \underbrace{\text{LIS'}}(8\ \ 6\ \ 3\ \ 6\ \ 9\ \ 7)$$

longest increasing sequence of the numbers starting with 8

Do we need to consider anything else for subsequences starting at 5?



1b: recursive solution

5 2 8 6 3 6 9 7

include 5 ↑

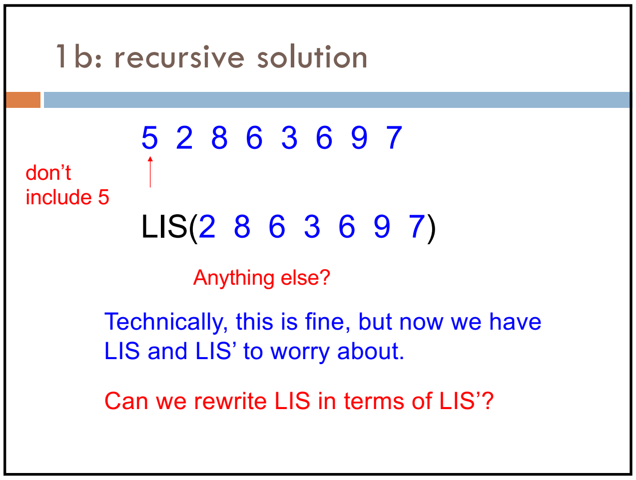
5 + LIS'(8 6 3 6 9 7)

5 + LIS'(6 3 6 9 7)

5 + LIS'(6 9 7)

5 + LIS'(9 7)

5 + LIS'(7)



1b: recursive solution

5 2 8 6 3 6 9 7

don't include 5 ↑

LIS(2 8 6 3 6 9 7)

Anything else?

Technically, this is fine, but now we have LIS and LIS' to worry about.

Can we rewrite LIS in terms of LIS'?

1b: recursive solution

$$LIS(X) = \max_i \{LIS'(i)\}$$

Longest increasing sequence for X
is the longest increasing sequence
starting at any element

And what is LIS' defined as (recursively)?

1b: recursive solution

$$LIS(X) = \max_i \{LIS'(i)\}$$

Longest increasing sequence for X
is the longest increasing sequence
starting at any element

$$LIS'(i) = 1 + \max_{j:i<j\leq n \text{ and } x_j>x_i} LIS'(j)$$

Longest increasing sequence starting at i

2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\leq n \text{ and } xj>xi} LIS'(j)$$

LIS' :

5 2 8 6 3 6 9 7

↑

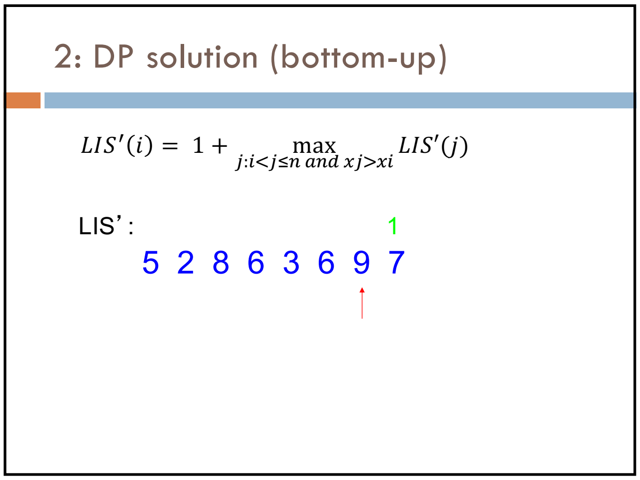
2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\leq n \text{ and } xj>xi} LIS'(j)$$

LIS' : 1

5 2 8 6 3 6 9 7

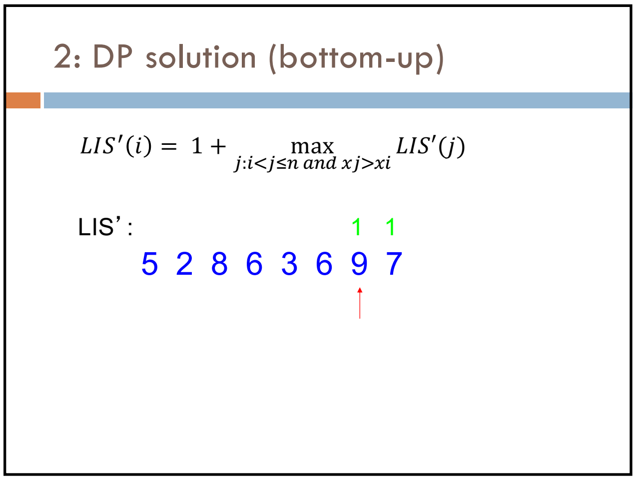
↑



2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\le n \text{ and } xj>xi} LIS'(j)$$

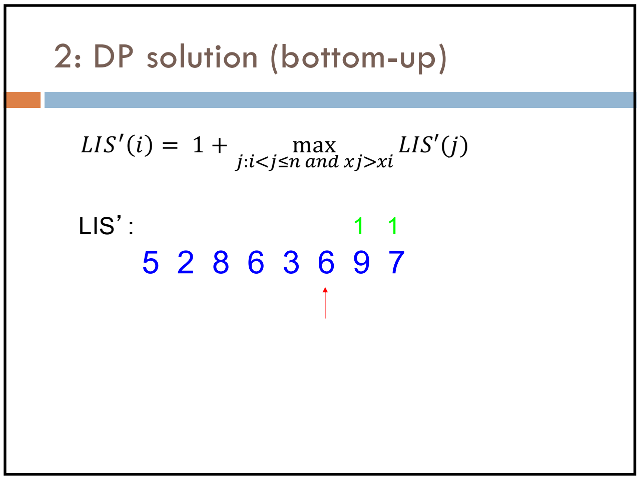
LIS' :								1
	5	2	8	6	3	6	9	7
							↑	



2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\le n \text{ and } xj>xi} LIS'(j)$$

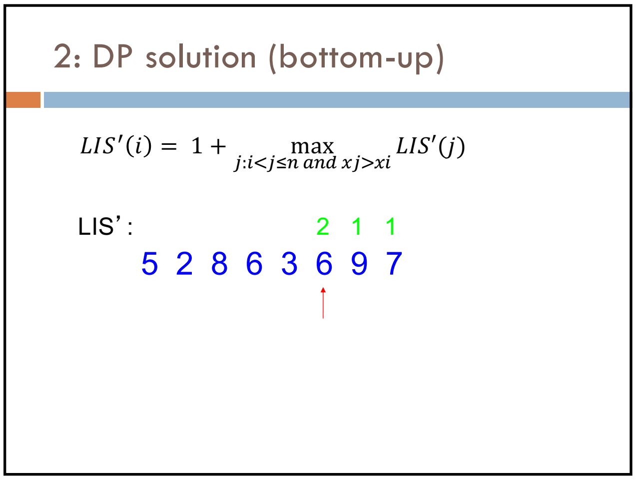
LIS' :							1	1
	5	2	8	6	3	6	9	7
							↑	



2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\le n \text{ and } xj>xi} LIS'(j)$$

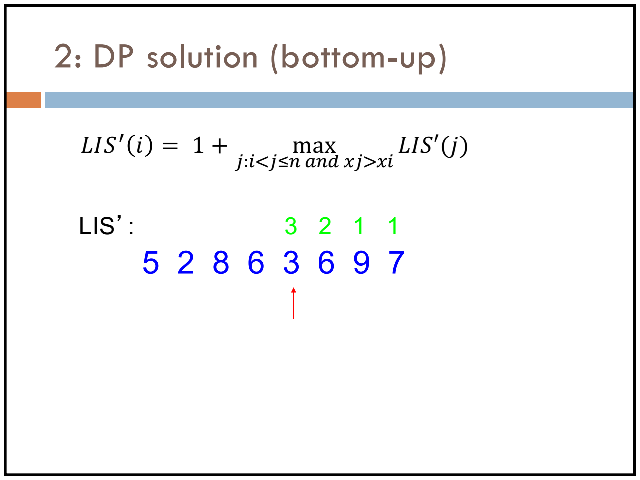
LIS' :							1	1
	5	2	8	6	3	6	9	7
						↑		



2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\le n \text{ and } xj>xi} LIS'(j)$$

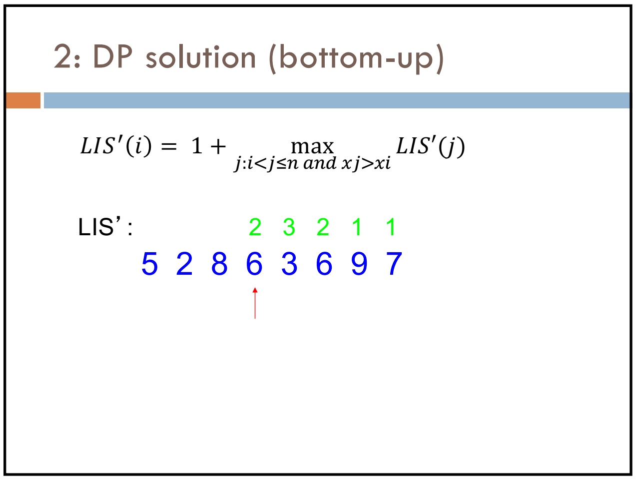
LIS' :						2	1	1
	5	2	8	6	3	6	9	7
						↑		



2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\le n \text{ and } xj>xi} LIS'(j)$$

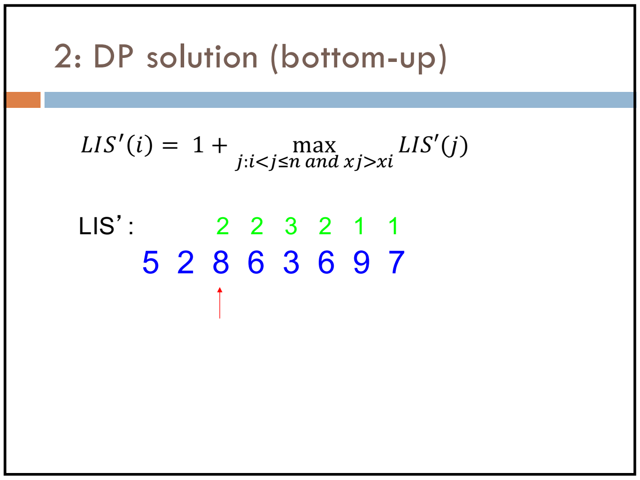
$$\begin{array}{lcccccccc} \text{LIS'}: & & & & & 3 & 2 & 1 & 1 \\ & 5 & 2 & 8 & 6 & 3 & 6 & 9 & 7 \\ & & & & & \uparrow & & & \end{array}$$



2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\le n \text{ and } xj>xi} LIS'(j)$$

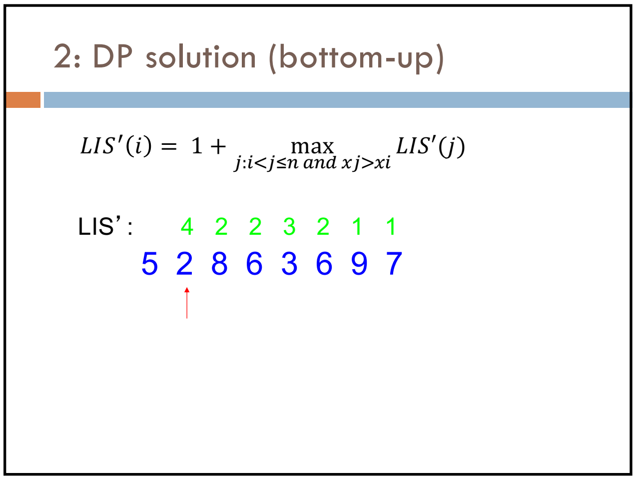
$$\begin{array}{lcccccccc} \text{LIS'}: & & & & 2 & 3 & 2 & 1 & 1 \\ & 5 & 2 & 8 & 6 & 3 & 6 & 9 & 7 \\ & & & & \uparrow & & & & \end{array}$$



2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\le n \text{ and } xj>xi} LIS'(j)$$

$$\begin{array}{lcccccccc} \text{LIS'}: & & & 2 & 2 & 3 & 2 & 1 & 1 \\ & 5 & 2 & 8 & 6 & 3 & 6 & 9 & 7 \\ & & & \uparrow & & & & & \end{array}$$



2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\le n \text{ and } xj>xi} LIS'(j)$$

$$\begin{array}{lcccccccc} \text{LIS'}: & & 4 & 2 & 2 & 3 & 2 & 1 & 1 \\ & 5 & 2 & 8 & 6 & 3 & 6 & 9 & 7 \\ & & \uparrow & & & & & & \end{array}$$

2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\leq n \text{ and } xj>xi} LIS'(j)$$

LIS’ : 3 4 2 2 3 2 1 1

5 2 8 6 3 6 9 7

2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\leq n \text{ and } xj>xi} LIS'(j)$$

LIS’ : 3 4 2 2 3 2 1 1

5 2 8 6 3 6 9 7

$$LIS(X) = \max_{i}\{LIS'(i)\}$$

2: DP solution (bottom-up)

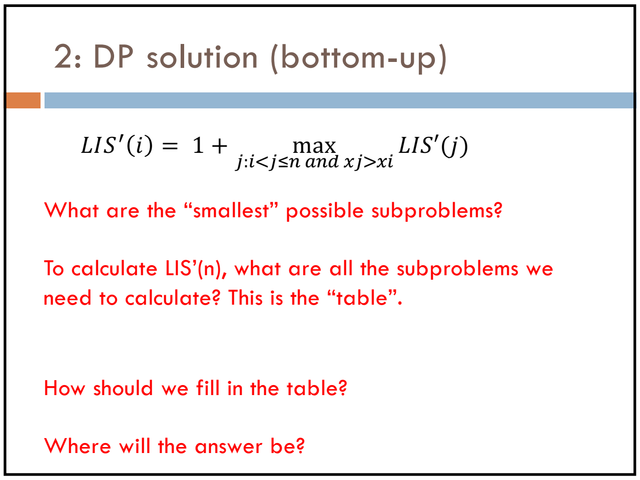
$$LIS'(i) = 1 + \max_{j:i<j\leq n \text{ and } xj>xi} LIS'(j)$$

What does the data structure for storing answers look like?

2: DP solution (bottom-up)

$$LIS'(i) = 1 + \max_{j:i<j\leq n \text{ and } xj>xi} LIS'(j)$$

1-D array: only one thing changes for recursive calls



2: DP solution (bottom-up)

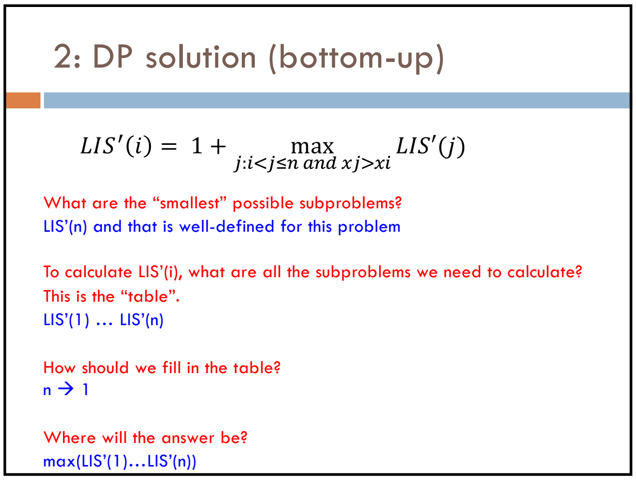
$$LIS'(i) = 1 + \max_{j:i<j\leq n \text{ and } xj>xi} LIS'(j)$$

What are the “smallest” possible subproblems?

To calculate LIS’(n), what are all the subproblems we need to calculate? This is the “table”.

How should we fill in the table?

Where will the answer be?



2: DP solution (bottom-up)

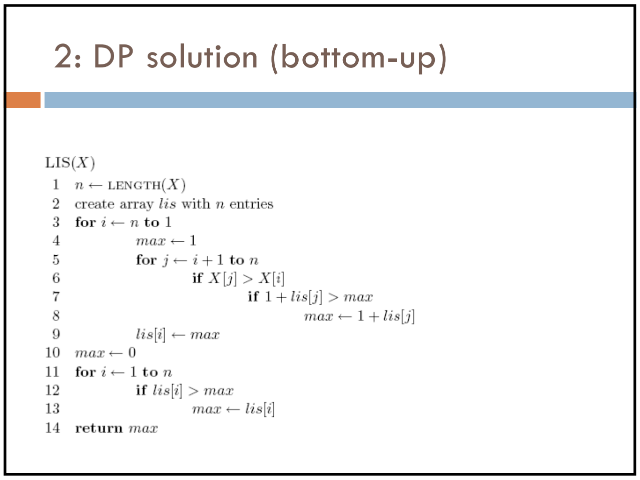
$$LIS'(i) = 1 + \max_{j:i<j\leq n \text{ and } xj>xi} LIS'(j)$$

What are the “smallest” possible subproblems?
LIS’(n) and that is well-defined for this problem

To calculate LIS’(i), what are all the subproblems we need to calculate? This is the “table”.
LIS’(1) … LIS’(n)

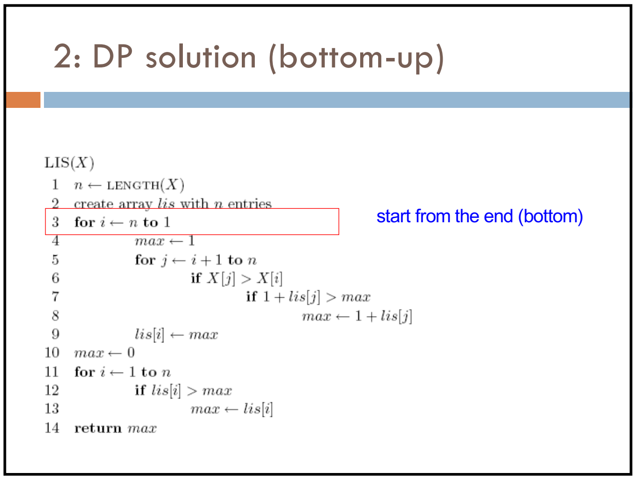
How should we fill in the table?
n → 1

Where will the answer be?
max(LIS’(1)…LIS’(n))



2: DP solution (bottom-up)

```
LIS($X$)
$n \leftarrow$ LENGTH($X$)
create array $lis$ with $n$ entries
for $i \leftarrow n$ to 1
    $max \leftarrow 1$
    for $j \leftarrow i+1$ to $n$
        if $X[j] > X[i]$
            if $1 + lis[j] > max$
                $max \leftarrow 1 + lis[j]$
    $lis[i] \leftarrow max$
$max \leftarrow 0$
for $i \leftarrow 1$ to $n$
    if $lis[i] > max$
        $max \leftarrow lis[i]$
return $max$
```



2: DP solution (bottom-up)

```
LIS($X$)
$n \leftarrow$ LENGTH($X$)
create array $lis$ with $n$ entries
for $i \leftarrow n$ to 1
    $max \leftarrow 1$
    for $j \leftarrow i+1$ to $n$
        if $X[j] > X[i]$
            if $1 + lis[j] > max$
                $max \leftarrow 1 + lis[j]$
    $lis[i] \leftarrow max$
$max \leftarrow 0$
for $i \leftarrow 1$ to $n$
    if $lis[i] > max$
        $max \leftarrow lis[i]$
return $max$
```

start from the end (bottom)

2: DP solution (bottom-up)

```
LIS($X$)
$n \leftarrow$ LENGTH($X$)
create array $lis$ with $n$ entries
for $i \leftarrow n$ to 1
    $max \leftarrow 1$
    for $j \leftarrow i+1$ to $n$
        if $X[j] > X[i]$
            if $1 + lis[j] > max$
                $max \leftarrow 1 + lis[j]$
    $lis[i] \leftarrow max$
$max \leftarrow 0$
for $i \leftarrow 1$ to $n$
    if $lis[i] > max$
        $max \leftarrow lis[i]$
return $max$
```

$$LIS'(i) = 1 + \max_{j: i<j\leq n \text{ and } xj>xi} LIS'(j)$$

2: DP solution (bottom-up)

```
LIS($X$)
$n \leftarrow$ LENGTH($X$)
create array $lis$ with $n$ entries
for $i \leftarrow n$ to 1
    $max \leftarrow 1$
    for $j \leftarrow i+1$ to $n$
        if $X[j] > X[i]$
            if $1 + lis[j] > max$
                $max \leftarrow 1 + lis[j]$
    $lis[i] \leftarrow max$
$max \leftarrow 0$
for $i \leftarrow 1$ to $n$
    if $lis[i] > max$
        $max \leftarrow lis[i]$
return $max$
```

$$\boxed{LIS(X) = \max_{i}\{LIS'(i)\}}$$

3: Analysis

```
LIS($X$)
$n \leftarrow$ LENGTH($X$)
create array $lis$ with $n$ entries
for $i \leftarrow n$ to 1
    $max \leftarrow 1$
    for $j \leftarrow i+1$ to $n$
        if $X[j] > X[i]$
            if $1 + lis[j] > max$
                $max \leftarrow 1 + lis[j]$
    $lis[i] \leftarrow max$
$max \leftarrow 0$
for $i \leftarrow 1$ to $n$
    if $lis[i] > max$
        $max \leftarrow lis[i]$
return $max$
```

Space requirements?

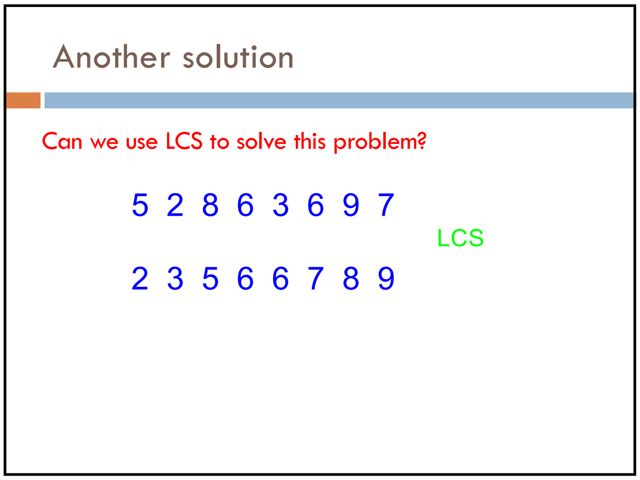
Running time?

3: Analysis

```
LIS($X$)
$n \leftarrow$ LENGTH($X$)
create array $lis$ with $n$ entries
for $i \leftarrow n$ to 1
    $max \leftarrow 1$
    for $j \leftarrow i+1$ to $n$
        if $X[j] > X[i]$
            if $1 + lis[j] > max$
                $max \leftarrow 1 + lis[j]$
    $lis[i] \leftarrow max$
$max \leftarrow 0$
for $i \leftarrow 1$ to $n$
    if $lis[i] > max$
        $max \leftarrow lis[i]$
return $max$
```

Space requirements: $\Theta(n)$

Running time: $\Theta(n^2)$



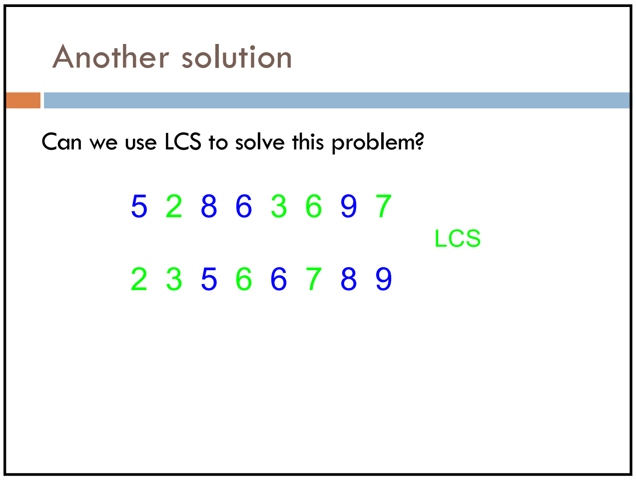
Another solution

Can we use LCS to solve this problem?

5 2 8 6 3 6 9 7

LCS

2 3 5 6 6 7 8 9



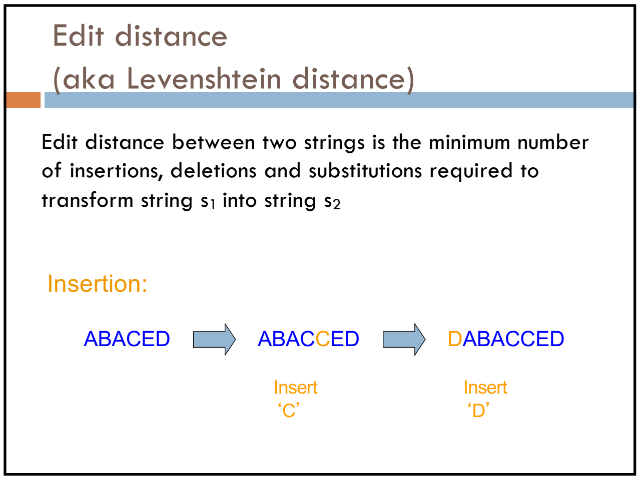
Another solution

Can we use LCS to solve this problem?

5 2 8 6 3 6 9 7

LCS

2 3 5 6 6 7 8 9



Edit distance
(aka Levenshtein distance)

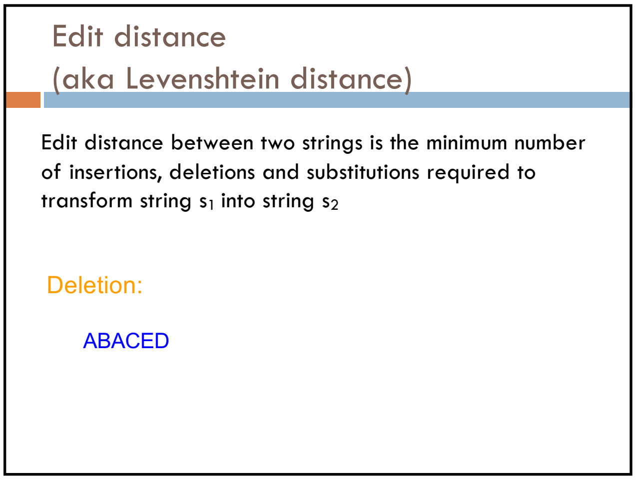
Edit distance between two strings is the minimum number of insertions, deletions and substitutions required to transform string s_1 into string s_2

Insertion:

ABACED → ABACCED → DABACCED

Insert 'C'

Insert 'D'



Edit distance
(aka Levenshtein distance)

Edit distance between two strings is the minimum number of insertions, deletions and substitutions required to transform string s_1 into string s_2

Deletion:

ABACED

Edit distance
(aka Levenshtein distance)

Edit distance between two strings is the minimum number of insertions, deletions and substitutions required to transform string s_1 into string s_2

Deletion:

ABACED → BACED

Delete 'A'

Edit distance
(aka Levenshtein distance)

Edit distance between two strings is the minimum number of insertions, deletions and substitutions required to transform string s_1 into string s_2

Deletion:

ABACED → BACED → BACE

Delete 'A'   Delete 'D'

Edit distance
(aka Levenshtein distance)

Edit distance between two strings is the minimum number of insertions, deletions and substitutions required to transform string s_1 into string s_2

Substitution:

ABACED → ABADED → ABADES

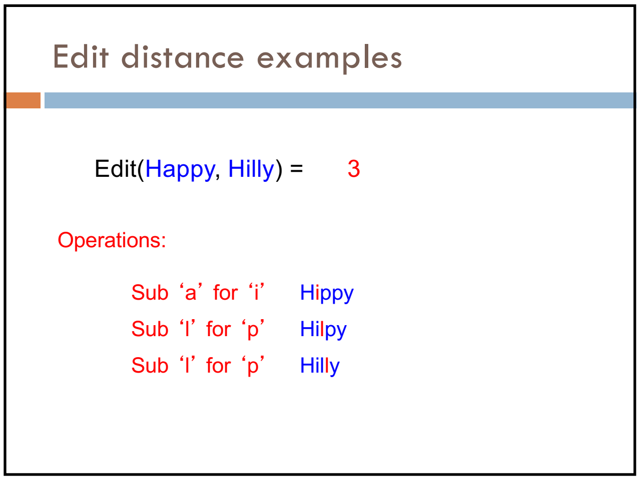
Sub 'D' for 'C'   Sub 'S' for 'D'

Edit distance examples

Edit(Kitten, Mitten) = 1

Operations:

Sub 'M' for 'K'   Mitten

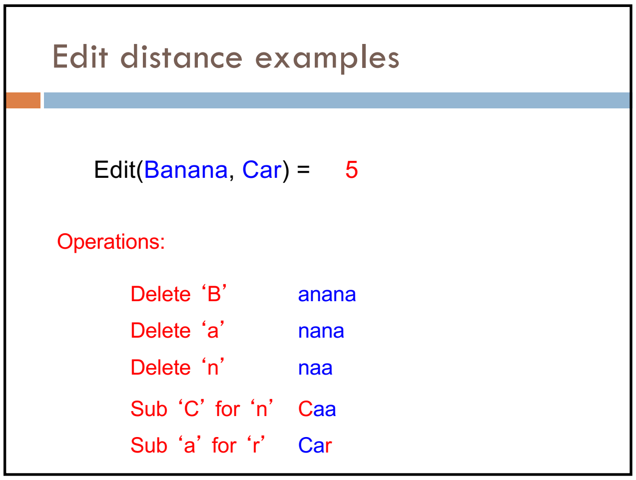


Edit distance examples

Edit(Happy, Hilly) = 3

Operations:

Sub 'a' for 'i'	Hippy
Sub 'l' for 'p'	Hilpy
Sub 'l' for 'p'	Hilly

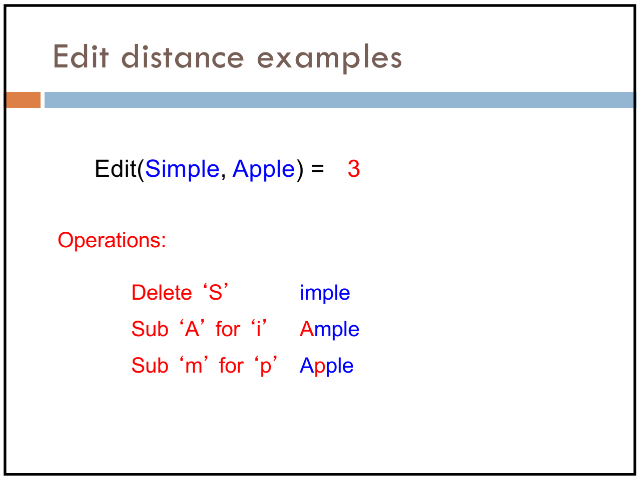


Edit distance examples

Edit(Banana, Car) = 5

Operations:

Delete 'B'	anana
Delete 'a'	nana
Delete 'n'	naa
Sub 'C' for 'n'	Caa
Sub 'a' for 'r'	Car

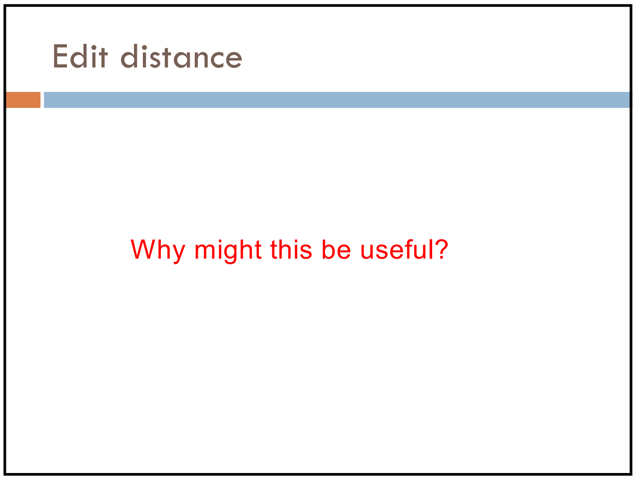


Edit distance examples

Edit(Simple, Apple) = 3

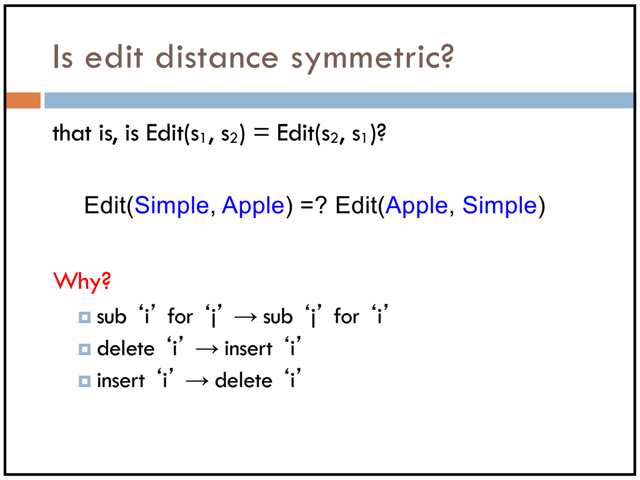
Operations:

Delete 'S'	imple
Sub 'A' for 'i'	Ample
Sub 'm' for 'p'	Apple



Edit distance

Why might this be useful?



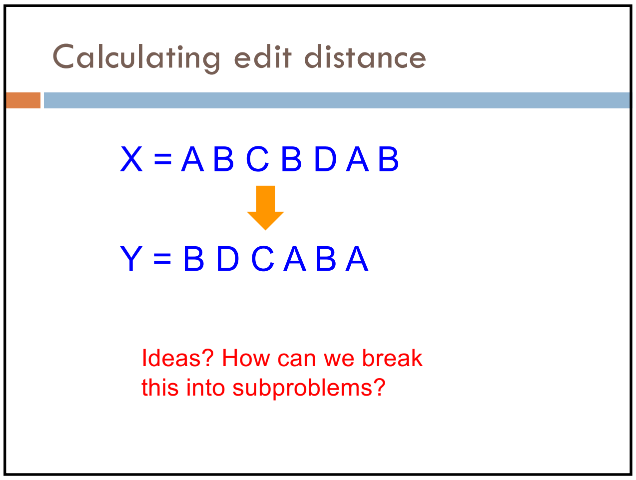
Is edit distance symmetric?

that is, is Edit(s_1, s_2) = Edit(s_2, s_1)?

Edit(Simple, Apple) =? Edit(Apple, Simple)

Why?

- sub ‘i’ for ‘j’ → sub ‘j’ for ‘i’
- delete ‘i’ → insert ‘i’
- insert ‘i’ → delete ‘i’

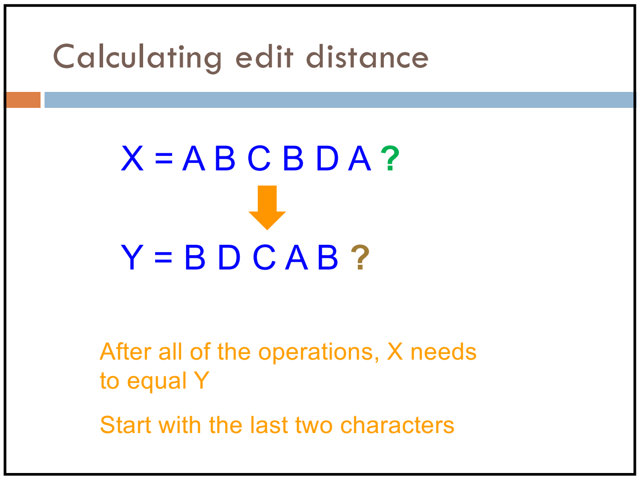


Calculating edit distance

X = A B C B D A B

Y = B D C A B A

Ideas? How can we break this into subproblems?



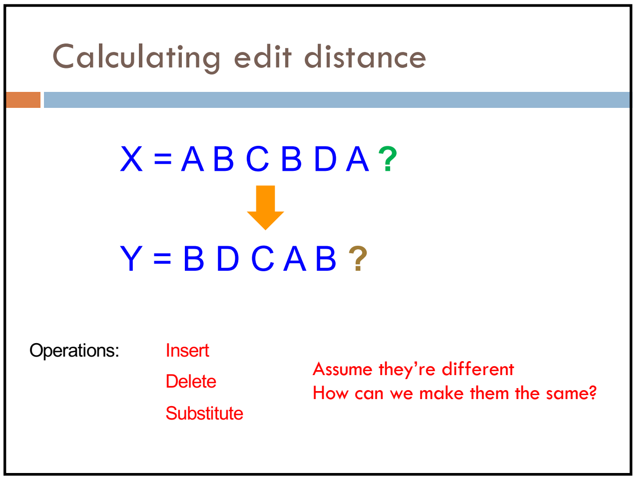
Calculating edit distance

X = A B C B D A ?

Y = B D C A B ?

After all of the operations, X needs to equal Y

Start with the last two characters



Calculating edit distance

X = A B C B D A ?

Y = B D C A B ?

Operations:

- Insert
- Delete
- Substitute

Assume they’re different
How can we make them the same?

Insert

X = A B C B D A ?

↓

Y = B D C A B ?

How can we use insert to transform X into Y?

Insert

X = A B C B D A ? ?

↓

Y = B D C A B ?

insert the last character of Y to the end of X

Insert

X = A B C B D A ? ?

↓

Y = B D C A B ?

How does this make the problem smaller?

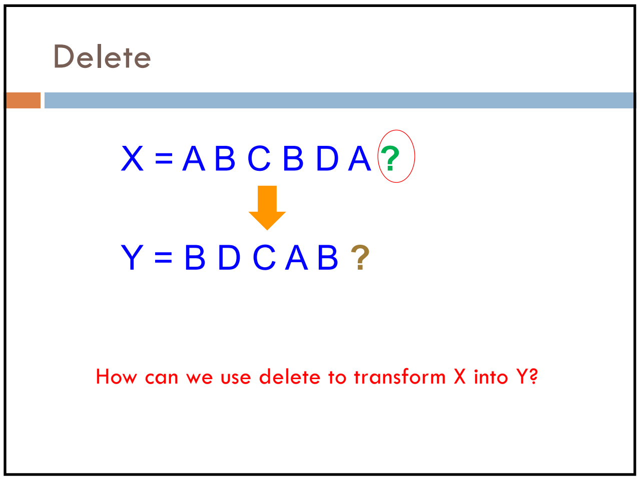
Insert

X = A B C B D A ? ?

Edit

Y = B D C A B ?

$$Edit(X,Y) = 1 + Edit(X_{1\ldots n}, Y_{1\ldots m-1})$$

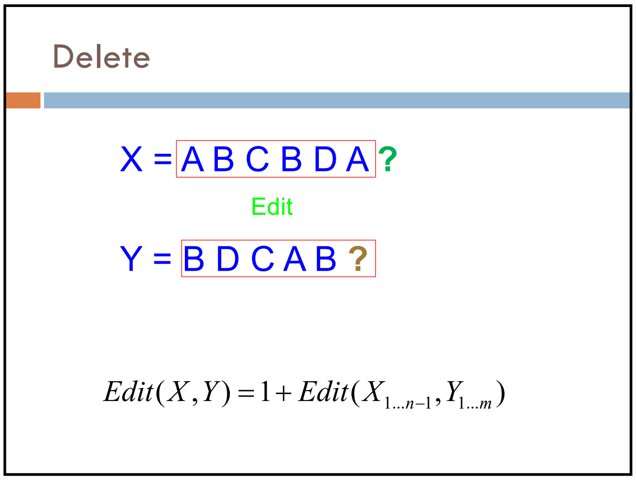


Delete

X = A B C B D A ?

Y = B D C A B ?

How can we use delete to transform X into Y?



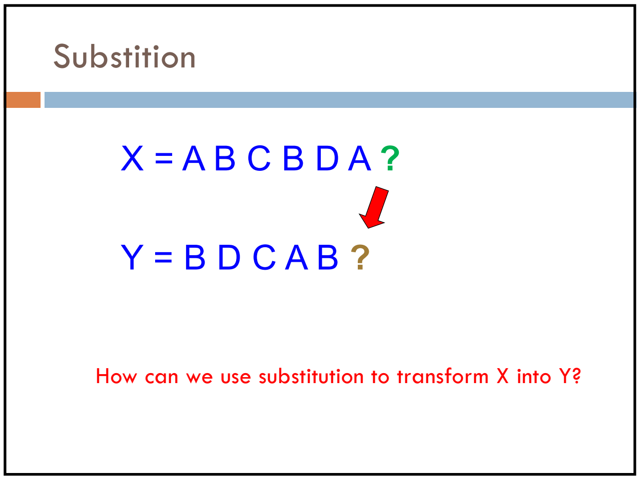
Delete

X = A B C B D A ?

Edit

Y = B D C A B ?

$$Edit(X,Y) = 1 + Edit(X_{1\ldots n-1}, Y_{1\ldots m})$$

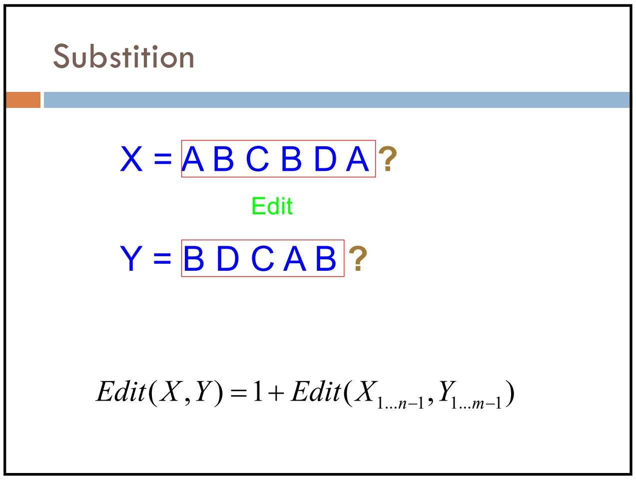


Substition

X = A B C B D A ?

Y = B D C A B ?

How can we use substitution to transform X into Y?



Substition

X = A B C B D A ?

Edit

Y = B D C A B ?

$$Edit(X,Y) = 1 + Edit(X_{1\ldots n-1}, Y_{1\ldots m-1})$$

Anything else?

X = A B C B D A ?

Y = B D C A B ?

Equal

X = A B C B D A ?

Y = B D C A B ?

What if the last characters are equal?

Equal

X = A B C B D A ?

Edit

Y = B D C A B ?

$$Edit(X,Y) = Edit(X_{1 \ldots n-1}, Y_{1 \ldots m-1})$$

1b: recursive solution - combining results

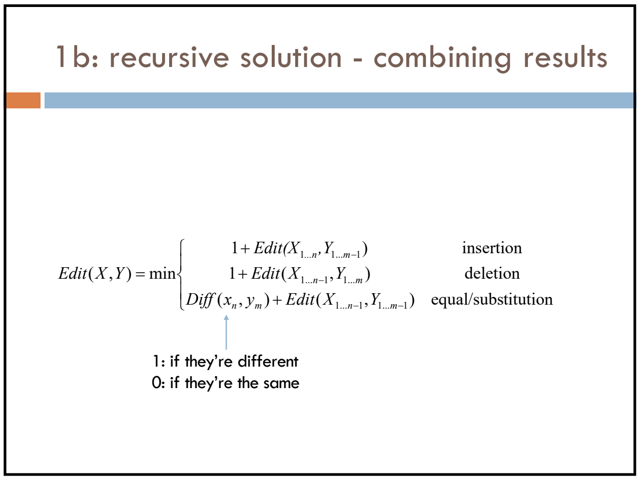
Insert: $Edit(X,Y) = 1 + Edit(X_{1 \ldots n}, Y_{1 \ldots m-1})$

Delete: $Edit(X,Y) = 1 + Edit(X_{1 \ldots n-1}, Y_{1 \ldots m})$

$X_n \neq Ym$
Substitute: $Edit(X,Y) = 1 + Edit(X_{1 \ldots n-1}, Y_{1 \ldots m-1})$

$X_n = Y_m$
Equal: $Edit(X,Y) = Edit(X_{1 \ldots n-1}, Y_{1 \ldots m-1})$

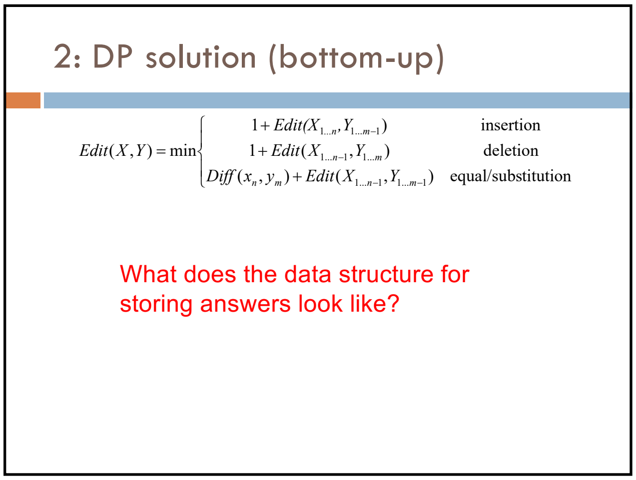
How do we decide between these?



1b: recursive solution - combining results

$$Edit(X,Y)=\min\begin{cases}1+Edit(X_{1\ldots n},Y_{1\ldots m-1}) & \text{insertion}\\ 1+Edit(X_{1\ldots n-1},Y_{1\ldots m}) & \text{deletion}\\ \underset{\uparrow}{Diff}(x_n,y_m)+Edit(X_{1\ldots n-1},Y_{1\ldots m-1}) & \text{equal/substitution}\end{cases}$$

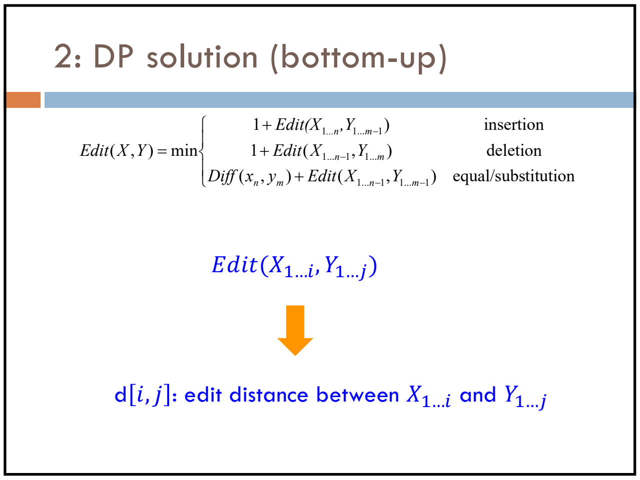
1: if they're different
0: if they're the same



2: DP solution (bottom-up)

$$Edit(X,Y)=\min\begin{cases}1+Edit(X_{1\ldots n},Y_{1\ldots m-1}) & \text{insertion}\\ 1+Edit(X_{1\ldots n-1},Y_{1\ldots m}) & \text{deletion}\\ Diff(x_n,y_m)+Edit(X_{1\ldots n-1},Y_{1\ldots m-1}) & \text{equal/substitution}\end{cases}$$

What does the data structure for storing answers look like?



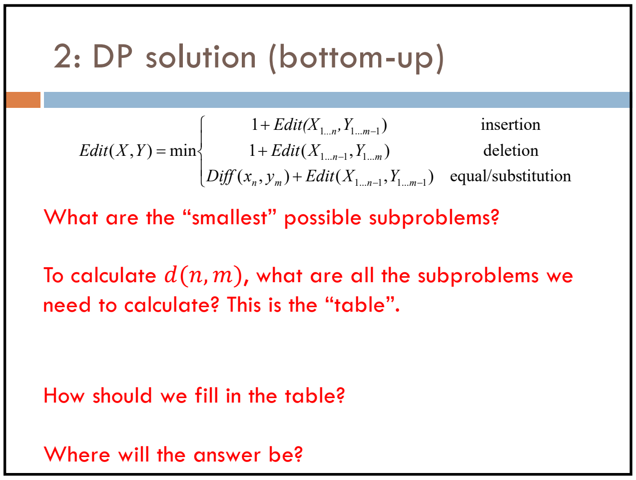
2: DP solution (bottom-up)

$$Edit(X,Y)=\min\begin{cases}1+Edit(X_{1\ldots n},Y_{1\ldots m-1}) & \text{insertion}\\ 1+Edit(X_{1\ldots n-1},Y_{1\ldots m}) & \text{deletion}\\ Diff(x_n,y_m)+Edit(X_{1\ldots n-1},Y_{1\ldots m-1}) & \text{equal/substitution}\end{cases}$$

$Edit(X_{1\ldots i},Y_{1\ldots j})$

$\Downarrow$

d$[i,j]$: edit distance between $X_{1\ldots i}$ and $Y_{1\ldots j}$



2: DP solution (bottom-up)

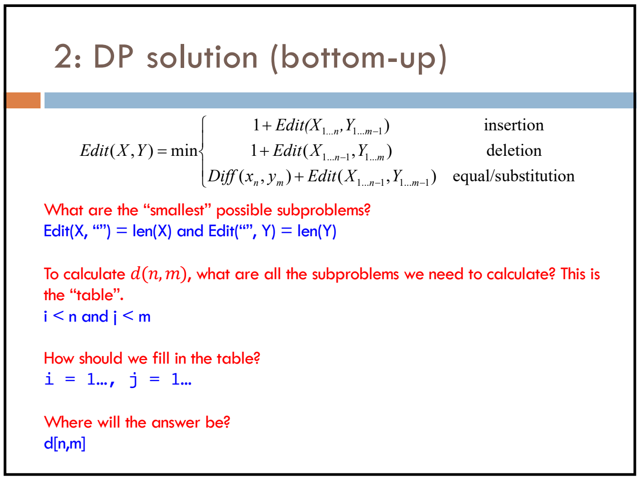
$$Edit(X,Y)=\min\begin{cases}1+Edit(X_{1\ldots n},Y_{1\ldots m-1}) & \text{insertion}\\ 1+Edit(X_{1\ldots n-1},Y_{1\ldots m}) & \text{deletion}\\ Diff(x_n,y_m)+Edit(X_{1\ldots n-1},Y_{1\ldots m-1}) & \text{equal/substitution}\end{cases}$$

What are the "smallest" possible subproblems?

To calculate $d(n,m)$, what are all the subproblems we need to calculate? This is the "table".

How should we fill in the table?

Where will the answer be?



2: DP solution (bottom-up)

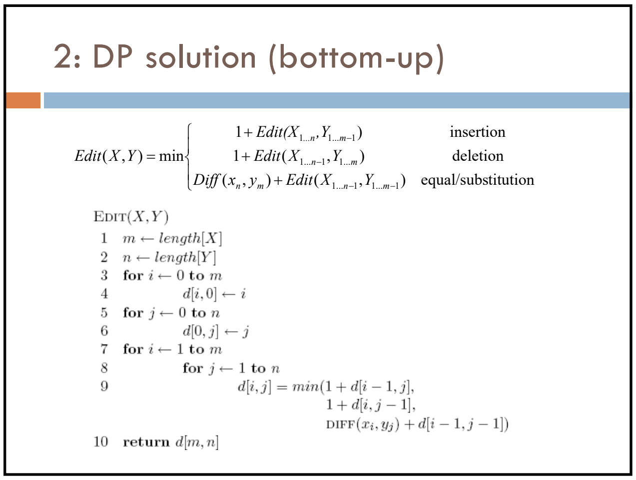
$$Edit(X,Y)=\min\begin{cases}1+Edit(X_{1\ldots n},Y_{1\ldots m-1}) & \text{insertion}\\ 1+Edit(X_{1\ldots n-1},Y_{1\ldots m}) & \text{deletion}\\ Diff(x_n,y_m)+Edit(X_{1\ldots n-1},Y_{1\ldots m-1}) & \text{equal/substitution}\end{cases}$$

What are the "smallest" possible subproblems?
Edit(X, "") = len(X) and Edit("", Y) = len(Y)

To calculate $d(n, m)$, what are all the subproblems we need to calculate? This is the "table".
i < n and j < m

How should we fill in the table?
`i = 1…, j = 1…`

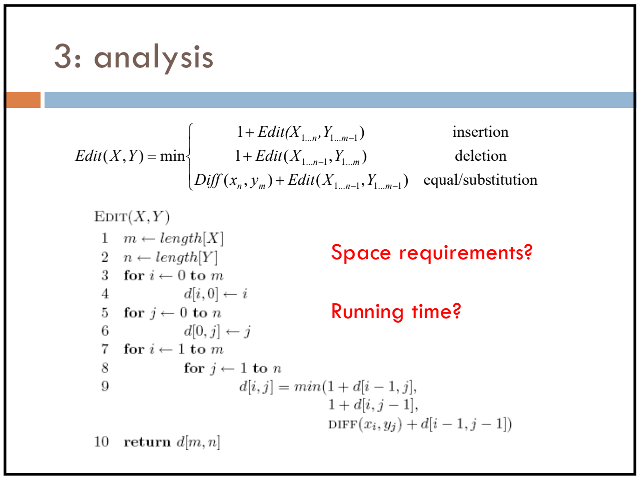
Where will the answer be?
d[n,m]



2: DP solution (bottom-up)

$$Edit(X,Y)=\min\begin{cases}1+Edit(X_{1\ldots n},Y_{1\ldots m-1}) & \text{insertion}\\ 1+Edit(X_{1\ldots n-1},Y_{1\ldots m}) & \text{deletion}\\ Diff(x_n,y_m)+Edit(X_{1\ldots n-1},Y_{1\ldots m-1}) & \text{equal/substitution}\end{cases}$$

```
EDIT($X, Y$)
$m \leftarrow length[X]$
$n \leftarrow length[Y]$
for $i \leftarrow 0$ to $m$
        $d[i, 0] \leftarrow i$
for $j \leftarrow 0$ to $n$
        $d[0, j] \leftarrow j$
for $i \leftarrow 1$ to $m$
        for $j \leftarrow 1$ to $n$
                $d[i, j] = min(1 + d[i-1, j],$
                              $1 + d[i, j-1],$
                              $\text{DIFF}(x_i, y_j) + d[i-1, j-1])$
return $d[m, n]$
```



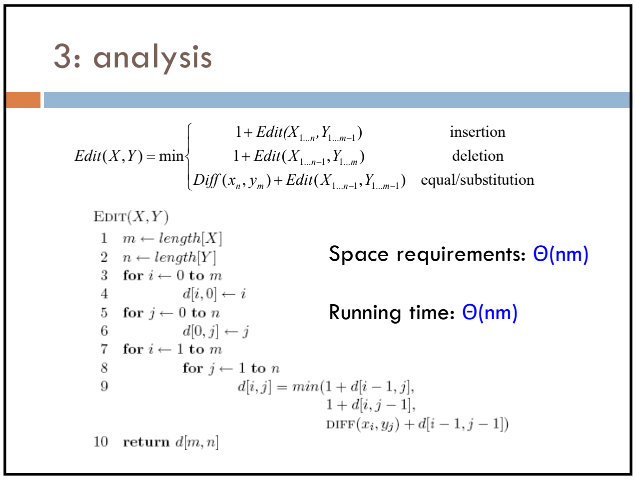
3: analysis

$$Edit(X,Y)=\min\begin{cases}1+Edit(X_{1\ldots n},Y_{1\ldots m-1}) & \text{insertion}\\ 1+Edit(X_{1\ldots n-1},Y_{1\ldots m}) & \text{deletion}\\ Diff(x_n,y_m)+Edit(X_{1\ldots n-1},Y_{1\ldots m-1}) & \text{equal/substitution}\end{cases}$$

```
EDIT($X, Y$)
$m \leftarrow length[X]$
$n \leftarrow length[Y]$
for $i \leftarrow 0$ to $m$
        $d[i, 0] \leftarrow i$
for $j \leftarrow 0$ to $n$
        $d[0, j] \leftarrow j$
for $i \leftarrow 1$ to $m$
        for $j \leftarrow 1$ to $n$
                $d[i, j] = min(1 + d[i-1, j],$
                              $1 + d[i, j-1],$
                              $\text{DIFF}(x_i, y_j) + d[i-1, j-1])$
return $d[m, n]$
```

Space requirements?

Running time?



3: analysis

$$Edit(X,Y)=\min\begin{cases}1+Edit(X_{1\ldots n},Y_{1\ldots m-1}) & \text{insertion}\\ 1+Edit(X_{1\ldots n-1},Y_{1\ldots m}) & \text{deletion}\\ Diff(x_n,y_m)+Edit(X_{1\ldots n-1},Y_{1\ldots m-1}) & \text{equal/substitution}\end{cases}$$

```
EDIT($X, Y$)
$m \leftarrow length[X]$
$n \leftarrow length[Y]$
for $i \leftarrow 0$ to $m$
        $d[i, 0] \leftarrow i$
for $j \leftarrow 0$ to $n$
        $d[0, j] \leftarrow j$
for $i \leftarrow 1$ to $m$
        for $j \leftarrow 1$ to $n$
                $d[i, j] = min(1 + d[i-1, j],$
                              $1 + d[i, j-1],$
                              $\text{DIFF}(x_i, y_j) + d[i-1, j-1])$
return $d[m, n]$
```

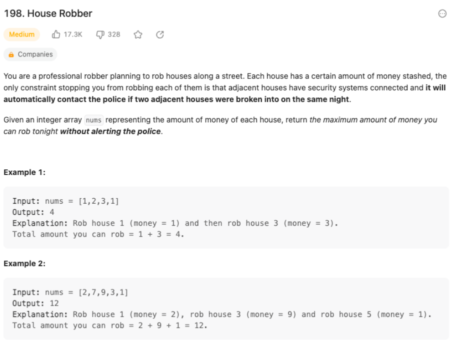
Space requirements: $\Theta(nm)$

Running time: $\Theta(nm)$

Edit distance variants

- Only include insertions and deletions
  - What does this do to substitutions?
- Include swaps, i.e. swapping two adjacent characters counts as one edit
- Weight insertion, deletion and substitution differently
- Weight **specific** character insertion, deletion and substitutions differently
- Length normalize the edit distance

https://leetcode.com/problems/house-robber/



198. House Robber

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***.

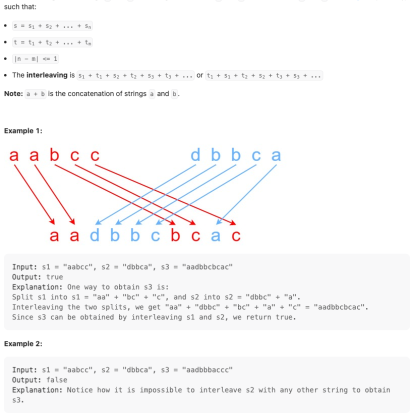
Example 1:

```
Input: nums = [1,2,3,1]
Output: 4
Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).
Total amount you can rob = 1 + 3 = 4.
```

Example 2:

```
Input: nums = [2,7,9,3,1]
Output: 12
Explanation: Rob house 1 (money = 2), rob house 3 (money = 9) and rob house 5 (money = 1).
Total amount you can rob = 2 + 9 + 1 = 12.
```

https://leetcode.com/problems/interleaving-string/description/



Given strings `s1`, `s2`, and `s3`, find whether `s3` is formed by an **interleaving** of `s1` and `s2`.

An **interleaving** of two strings `s` and `t` is a configuration where `s` and `t` are divided into `n` and `m` substrings respectively, such that:

- $s = s_1 + s_2 + ... + s_n$
- $t = t_1 + t_2 + ... + t_m$
- `|n - m| <= 1`
- The **interleaving** is $s_1 + t_1 + s_2 + t_2 + s_3 + t_3 + ...$ or $t_1 + s_1 + t_2 + s_2 + t_3 + s_3 + ...$

Note: `a + b` is the concatenation of strings `a` and `b`.

Example 1:

```
Input: s1 = "aabcc", s2 = "dbbca", s3 = "aadbbcbcac"
Output: true
Explanation: One way to obtain s3 is:
Split s1 into s1 = "aa" + "bc" + "c", and s2 into s2 = "dbbc" + "a".
Interleaving the two splits, we get "aa" + "dbbc" + "bc" + "a" + "c" = "aadbbcbcac".
Since s3 can be obtained by interleaving s1 and s2, we return true.
```

Example 2:

```
Input: s1 = "aabcc", s2 = "dbbca", s3 = "aadbbbaccc"
Output: false
Explanation: Notice how it is impossible to interleave s2 with any other string to obtain
s3.
```