CPSC 420 Lecture 14: Today's announcements:

- Examlet 2 on Feb 17 in class. Closed book & no notes
- Reading: Ch.3 Dynamic Programming [by Erickson]
  Introduction to Algorithms [by Manber]
  An $O(ND)$ difference algorithm and its variations [by Myers]
  `https://go.exlibris.link/Wv9Rf8Tn`

Today's Plan

- Dynamic programming
  - Longest Increasing Subsequence
  - Edit Distance

Best Increasing Subsequences Dynamic Programming

$$8 \quad 3 \quad 4 \quad 9 \quad 6 \quad 2 \quad 1 \quad 5 \quad 7 \quad 2$$

$R[k]$ **extends** $\text{BIS}[j]$ iff $\text{BIS}[j].\mathit{last} < R[k]$ and
$(j = |\text{BIS}| \text{ or } \text{BIS}[j+1].\mathit{last} > R[k])$

Maintain lists $L[0], L[1], ..$ where $L_0[0] = [0]$. After we scan $R[1..k]$

$$L_k[j] = \begin{cases} L_{k-1}[j] \circ R[k] & \text{if } R[k] \text{ extends } \text{BIS}_{k-1}[j] \\ L_{k-1}[j] & \text{otherwise} \end{cases}$$

$$\begin{aligned} L[0] &= \quad 0 \\ L[1] &= \quad 8 \quad 3 \quad 2 \quad 1 \\ L[2] &= \quad 4 \quad 2 \\ L[3] &= \quad 9 \quad 6 \quad 5 \\ L[4] &= \quad 7 \end{aligned}$$

a b means b extended BIS ending with a

use binary search to find j

Running time?

As a result list $L_k[i]$ ends with $\text{BIS}_k[j].\mathit{last}$ at step k.

Use LIS to solve LCS [Hunt & Szymanski, McIlroy 1977]

		A	B	C	B	A	C	C	B	$= X$
	B		x		x				x	
	C			x			x	x		
	D									
$Y =$	A	x				x				
	B		x		x				x	
	C			x			x	x		
	C			x			x	x		

Use LIS to solve LCS [Hunt & Szymanski, McIlroy 1977]

	A	B	C	B	A	C	C	B	$= X$
B		x		x				x	
C			x			x	x		
D									
$Y =$ A	x				x				
B		x		x				x	
C			x			x	x		
C			x			x	x		

1 , 2
2 , 3
5 , 4
6 , 6
7 , 7

Longest Common Subsequence = BCBCC

What properties do the circled entries have?

Use LIS to solve LCS [Hunt & Szymanski, McIlroy 1977]

X = A B C B A C C B

Y = B C D A B C C

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

1 , 2
2 , 3
5 , 4
6 , 6
7 , 7

Longest Common Subsequence = BCBCC

What properties do the circled entries have?

They form a sequence of index-pairs of matches that increase in both dimensions.

Idea: Use LIS to find a longest such "doubly-increasing" sequence in the sequence of all index-pairs of matches.

Use LIS to solve LCS [Hunt & Szymanski, McIlroy 1977]

X = A B C B A C C B

Y	A	B	C	B	A	C	C	B
B		x		x				x
C			x			x	x	
D								
A	x				x			
B		x		x				x
C			x			x	x	
C			x			x	x	

1 , 2
2 , 3
5 , 4
6 , 6
7 , 7

Longest Common Subsequence = BCBCC

Write index-pairs of matches in top-bottom, right-left order.

Y →	1	1	1	2	2	2	4	4	5	5	5	6	6	6	7	7	7	
X →	8	4	2	7	6	3	5	1	8	4	2	7	6	3	7	6	3	$= S$

A common sequence of X and Y corresponds to an increasing sequence of S (and vice-versa).

Use LIS to solve LCS [Hunt & Szymanski, McIlroy 1977]

LCS(X, Y)
1. Stably sort X keeping track of each character's index in X
2. for $i = 1$ to n
3. look up $Y[i]$ in sorted X
4. add indices of matching characters in reverse order to S
5. $Q = \text{LIS}(S)$
6. Output $X[Q]$ (characters in X indexed by Q)

$X =$ A B C B A C C B $\qquad Y =$ B C D A B C C

stably sort

A	A	B	B	B	C	C	C
1	5	2	4	8	3	6	7

8 4 2 6 3

time $O(m \log m) + O(n \log m) + |S| + O(|S| \log |S|)$

$H[A] = 1\ 5$

$H[B] = 2\ 4\ 8$

$H[C] = 3\ 6\ 7$

$H[D] = \emptyset$

In practice: Use hashing rather than sorting.

A different approach to edit distance [Myers 1986]

The **edit distance** between X and Y is the minimum number of **inserts** and **deletes** to transform X into Y.

Problem: Given two strings X and Y, find edit distance $d(X, Y)$.

elephant $\xrightarrow{d}$ elephnt $\xrightarrow{i}$ telephnt $\xrightarrow{d}$ telephn $\xrightarrow{i}$ telephon $\xrightarrow{i}$ telephone

LCS(elephant, telephone) $= \underbrace{\text{elephn}}_{6}$

$|X| + |Y| - 2LCS(X,Y) \geq d(X,Y)$

$\leq$

A different approach to edit distance [Myers 1986]

The **edit distance** between X and Y is the minimum number of **inserts** and **deletes** to transform X into Y.

Problem: Given two strings X and Y, find edit distance $d(X, Y)$.

elephant $\xrightarrow{d}$ elephnt $\xrightarrow{i}$ telephnt $\xrightarrow{d}$ telephn $\xrightarrow{i}$ telephon $\xrightarrow{i}$ telephone

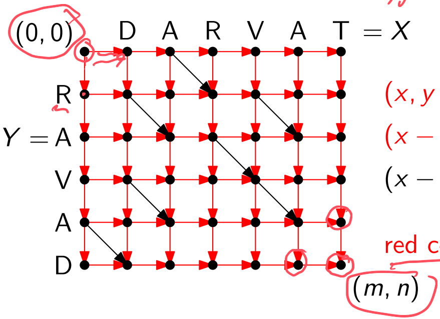
LCS(elephant, telephone) = elephn

$d(X, Y) = |X| + |Y| - 2|\mathrm{LCS}(X, Y)|$

So why are we talking about edit distance?

A different approach to edit distance [Myers 1986]

(i, j) vertex X[1 … i] into Y[1 … j]



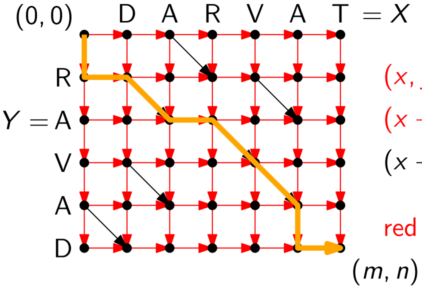
$(x, y-1) \downarrow (x, y)$ insert $Y[y]$

$(x-1, y) \rightarrow (x, y)$ delete $X[x]$

$(x-1, y-1) \searrow (x, y)$ $X[x] = Y[y]$

red cost = 1   black cost = 0

A different approach to edit distance [Myers 1986]



$(x, y-1) \downarrow (x, y)$ insert $Y[y]$

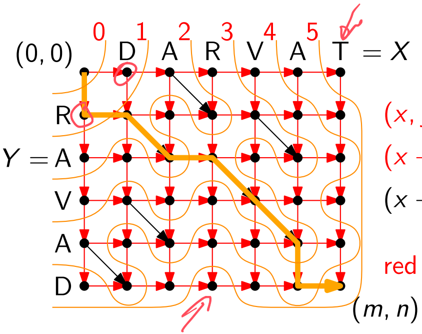
$(x-1, y) \rightarrow (x, y)$ delete $X[x]$

$(x-1, y-1) \searrow (x, y)$ $X[x] = Y[y]$

red cost = 1 black cost = 0

Edit distance $d(X, Y) \equiv$ shortest path length from $(0, 0)$ to (m, n).

A different approach to edit distance [Myers 1986]



$(x, y-1) \downarrow (x, y)$ insert $Y[y]$

$(x-1, y) \rightarrow (x, y)$ delete $X[x]$

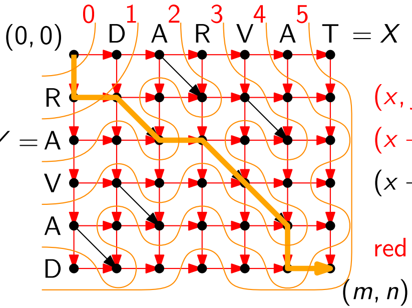
$(x-1, y-1) \searrow (x, y)$ $X[x] = Y[y]$

red cost $= 1$ black cost $= 0$

Edit distance $d(X, Y) \equiv$ shortest path length from $(0, 0)$ to (m, n).

Dijkstra's shortest path alg. explores at most $(2d+1)(n+m)$ vertices.

A different approach to edit distance [Myers 1986]



$(x, y-1) \downarrow (x, y)$ insert $Y[y]$

$(x-1, y) \rightarrow (x, y)$ delete $X[x]$

$(x-1, y-1) \searrow (x, y)$ $X[x] = Y[y]$

red cost $= 1$ black cost $= 0$

Edit distance $d(X, Y) \equiv$ shortest path length from $(0, 0)$ to (m, n).

Dijkstra's shortest path alg. explores at most $(2d + 1)(n + m)$ vertices.

Running time: $O(d(n + m))$.