14.2.5
Reducing space for edit distance

Matrix and DAG of computation of edit distance

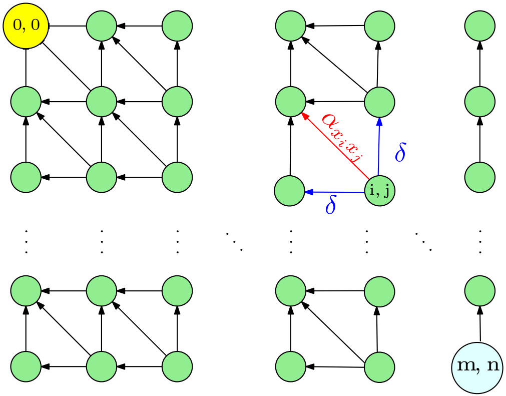

Figure: Iterative algorithm in previous slide computes values in row order.

Optimizing Space

1. Recall
$$M(i,j) = \min \begin{cases} \alpha_{x_i y_j} + M(i-1, j-1), \\ \delta + M(i-1, j), \\ \delta + M(i, j-1) \end{cases}$$

2. Entries in jth column only depend on $(j-1)$st column and earlier entries in jth column
3. Only store the current column and the previous column reusing space; $N(i,0)$ stores $M(i,j-1)$ and $N(i,1)$ stores $M(i,j)$

Example: DEED vs. BREAD filled by column

	ε	***D***	***R***	***E***	***A***	***D***
ε						
D						
E						
E						
D						

Example: DEED vs. BREAD filled by column

	ε	***D***	***R***	***E***	***A***	***D***
ε	0	1	2	3	4	5
D	1					
E	2					
E	3					
D	3					

Example: DEED vs. BREAD filled by column

	ε	***D***	***R***	***E***	***A***	***D***
ε	0	1	2	3	4	5
D	1	0				
E	2	1				
E	3	2				
D	3	3				

Example: DEED vs. BREAD filled by column

	ε	***D***	***R***	***E***	***A***	***D***
ε	0	1	2	3	4	5
D	1	0	1			
E	2	1	1			
E	3	2	2			
D	3	3	3			

Example: DEED vs. BREAD filled by column

	ε	***D***	***R***	***E***	***A***	***D***
ε	0	1	2	3	4	5
D	1	0	1	2		
E	2	1	1	1		
E	3	2	2	1		
D	3	3	3	2		

Example: DEED vs. BREAD filled by column

	ε	***D***	***R***	***E***	***A***	***D***
ε	0	1	2	3	4	5
D	1	0	1	2	3	
E	2	1	1	1	2	
E	3	2	2	1	2	
D	3	3	3	2	2	

Example: DEED vs. BREAD filled by column

	ε	***D***	***R***	***E***	***A***	***D***
ε	0	1	2	3	4	5
D	1	0	1	2	3	4
E	2	1	1	1	2	3
E	3	2	2	1	2	3
D	3	3	3	2	2	2

Computing in column order to save space

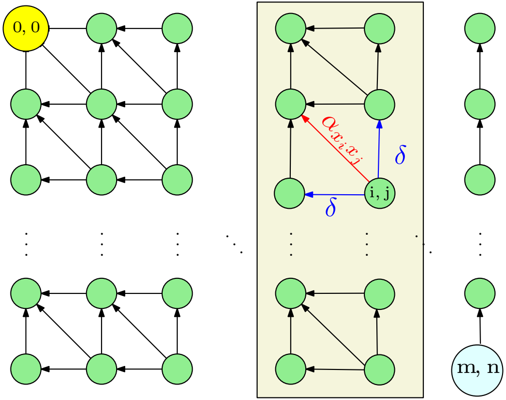

Figure: $\boldsymbol{M(i,j)}$ only depends on previous column values. Keep only two columns and compute in column order.

Space Efficient Algorithm

```
for all i do N[i,0] = iδ
for j = 1 to n do
    N[0,1] = jδ (* corresponds to M(0,j) *)
    for i = 1 to m do
```

$$N[i,1] = \min \begin{cases} \alpha_{x_i y_j} + N[i-1,0] \\ \delta + N[i-1,1] \\ \delta + N[i,0] \end{cases}$$

```
    for i = 1 to m do
        Copy N[i,0] = N[i,1]
```

Analysis

Running time is $O(mn)$ and space used is $O(2m) = O(m)$

Analyzing Space Efficiency

1. From the $m \times n$ matrix M we can construct the actual alignment (exercise)
2. Matrix N computes cost of optimal alignment but no way to construct the actual alignment
3. Space efficient computation of alignment? More complicated algorithm — see notes and Kleinberg-Tardos book.

THE END

...

(for now)