Dynamic Programming
CSCI 432

Edit Distance

Given two strings, how many edits are needed to turn one string into another?

Input:

- Two strings
- Cost function
  (e.g., non-matches = +1)

S – N O W Y
S U N N – Y

cost = 3

Edit Distance

We want to align two strings, $x = [x_1, \ldots, x_n]$ and $y = [y_1, \ldots, y_m]$.

$E(i, j)$ = optimal cost of aligning $[x_1, \ldots, x_i]$ and $[y_1, \ldots, y_j]$.

Can we say anything about optimal alignment of $[x_1, \ldots, x_i]$ and $[y_1, \ldots, y_j]$?

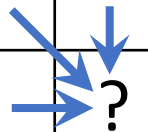
Alignment	Cost
$\begin{matrix} x_i \\ y_j \end{matrix}$	$\{0,1\} + E(i-1, j-1)$
$\begin{matrix} x_i \\ - \end{matrix}$	$1 + E(i-1, j)$
$\begin{matrix} - \\ y_j \end{matrix}$	$1 + E(i, j-1)$

$$E(i,j) = \min \begin{cases} \text{diff}(i,j) + E(i-1, j-1) \\ 1 + E(i-1, j) \\ 1 + E(i, j-1) \end{cases}$$

$$\text{where diff}(i,j) = \begin{cases} 0, & x_i = y_j \\ 1, & x_i \neq y_j \end{cases}$$

Edit Distance

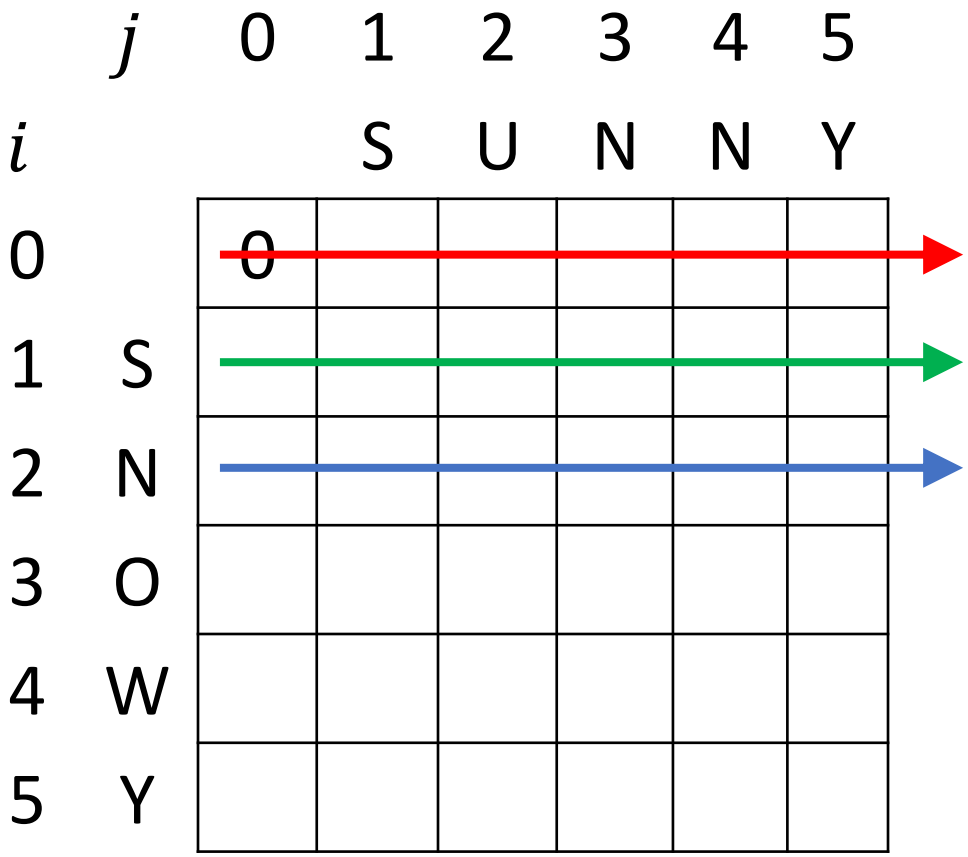
i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0					
1	S						
2	N						
3	O				?		
4	W						
5	Y						



$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & x[i] \neq y[j] \end{cases}$$

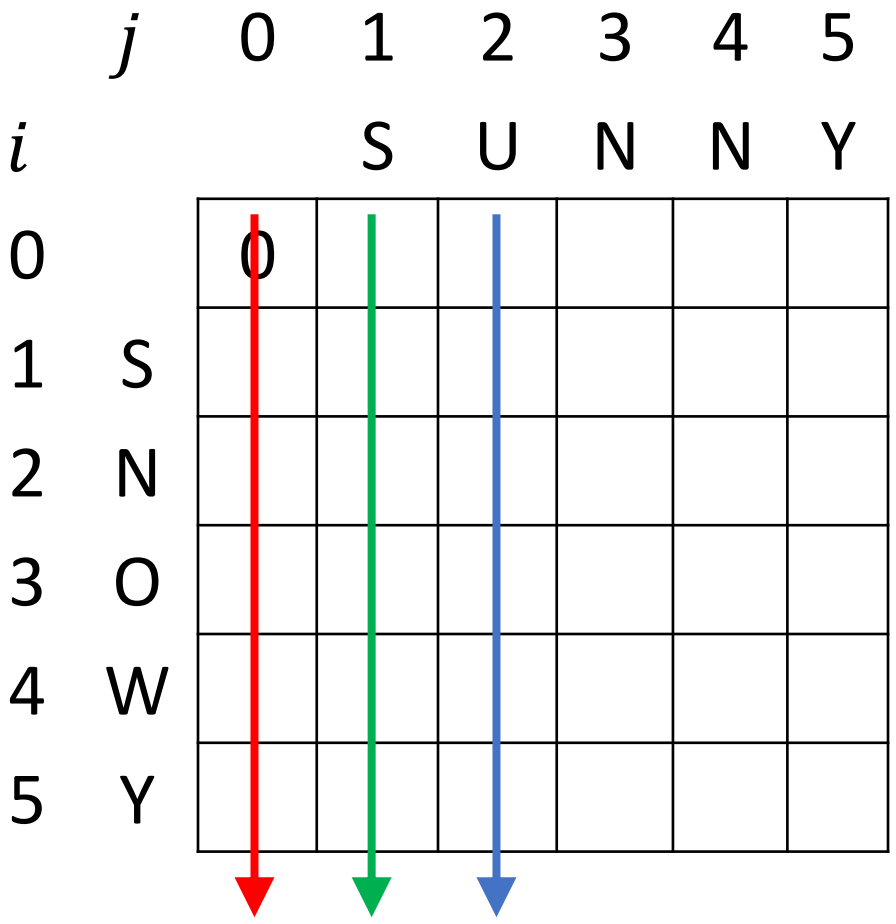
Edit Distance



$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & x[i] \neq y[j] \end{cases}$$

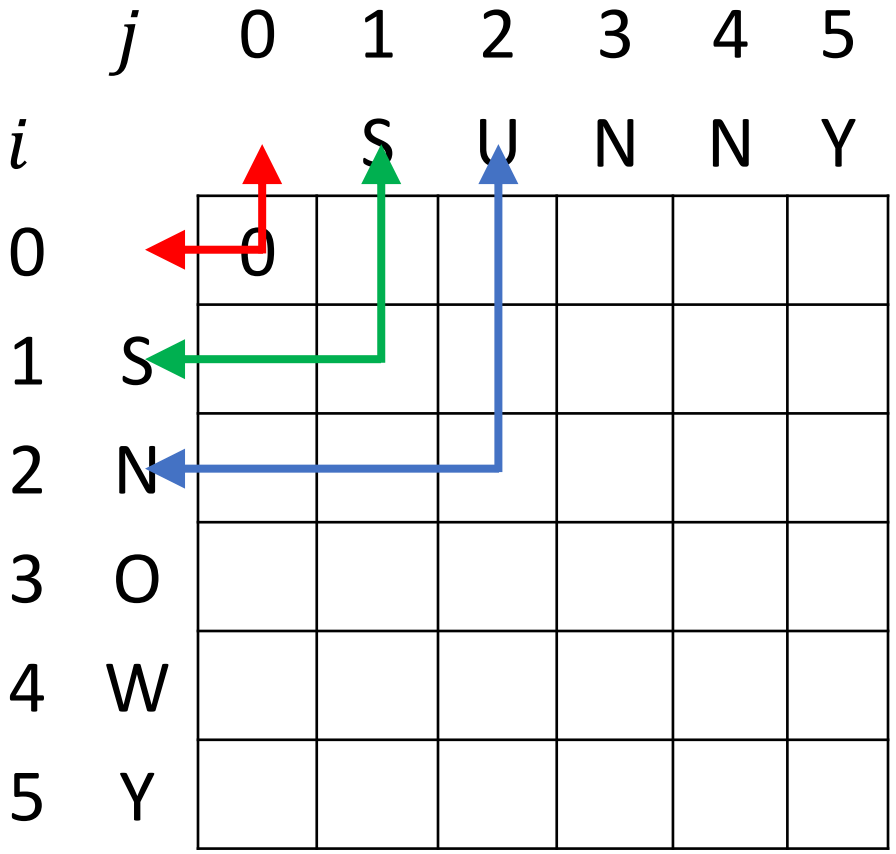
Edit Distance



$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & x[i] \neq y[j] \end{cases}$$

Edit Distance



$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & x[i] \neq y[j] \end{cases}$$

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Running Time?

Fill out $n \times m$ table with constant operations: $O(nm)$

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Diagonal move indicates match.

Vertical move indicates space inserted in j.

Horizontal move indicates space inserted in i.

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Diagonal move indicates match.

Vertical move indicates space inserted in j.

Horizontal move indicates space inserted in i.

S – N O W Y

S U N N – Y

Edit Distance

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Diagonal move indicates match.

Vertical move indicates space inserted in j.

Horizontal move indicates space inserted in i.

S – N O W Y

S U N – N Y

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Diagonal move indicates match.

Vertical move indicates space inserted in j.

Horizontal move indicates space inserted in i.

S N O W Y

S U N N Y

Edit Distance

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

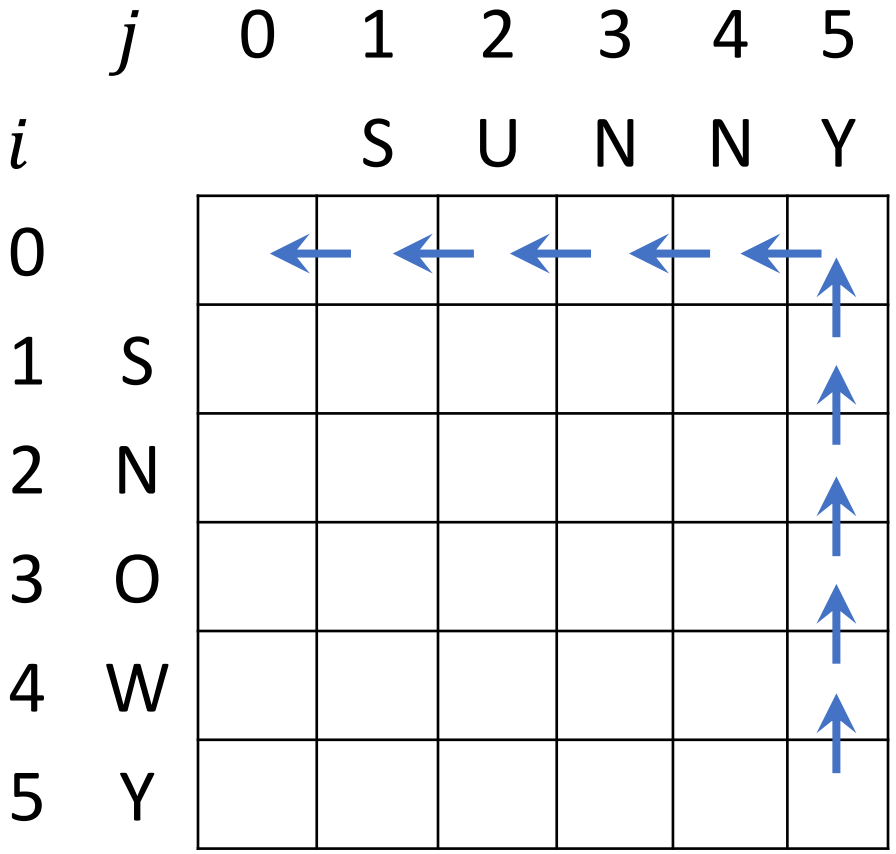
Diagonal move indicates match.

Vertical move indicates space inserted in j.

Horizontal move indicates space inserted in i.

Running time for backtracking?

Edit Distance



Diagonal move indicates match.

Running time for backtracking?

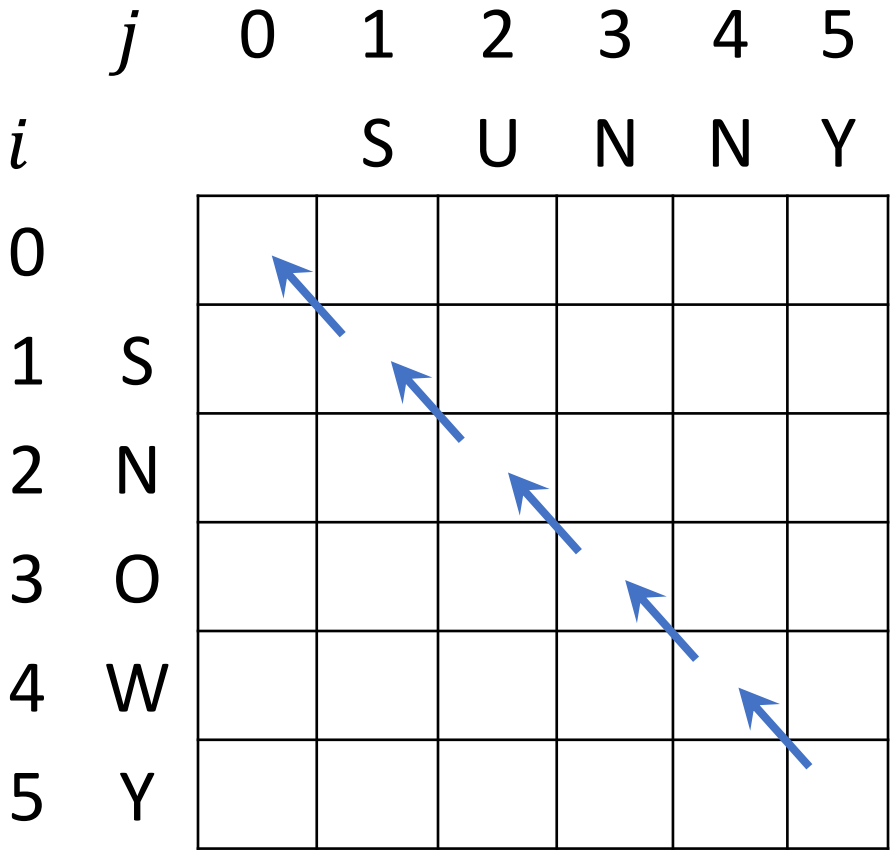
At worst, only match to insertions:

$$O(m+n) = O(\max(n,m))$$

_ _ _ _ _ S N O W Y

S U N N Y _ _ _ _ _

Edit Distance



Diagonal move indicates match.

Running time for backtracking?

At worst, only match to insertions:

$$O(m+n) = O(\max(n,m))$$

At best, no insertions in larger string:

$$O(\max(n,m))$$

S N O W Y

S U N N Y

Edit Distance

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Diagonal move indicates match.

Running time for backtracking?

$$O(\max(n, m))$$

At worst, only match to insertions:

$$O(m + n) = O(\max(n, m))$$

At best, no insertions in larger string:

$$O(\max(n, m))$$

Edit Distance

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Edit Distance

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3



$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Y

Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Y

Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Y

Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i		j 0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Y
Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

Y

Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

W Y

N Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

W Y

N Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

W Y

N Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

W Y

N Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

O	W	Y
–	N	Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

N O W Y

N – N Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

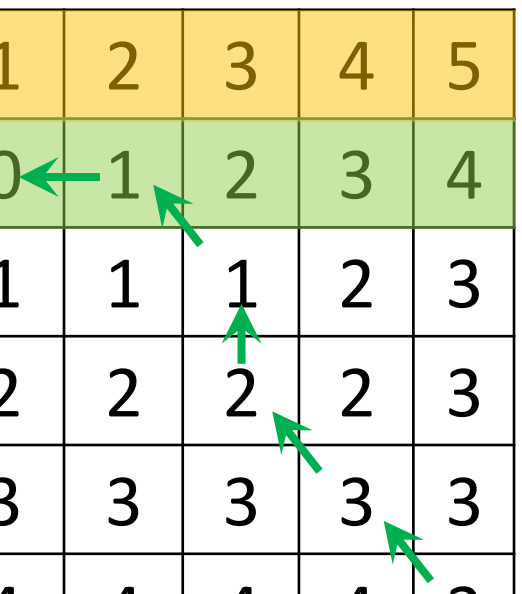
Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3



–	N	O	W	Y
U	N	–	N	Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i		j	0	1	2	3	4	5
				S	U	N	N	Y
0			0	1	2	3	4	5
1	S		1	0	1	2	3	4
2	N		2	1	1	1	2	3
3	O		3	2	2	2	2	3
4	W		4	3	3	3	3	3
5	Y		5	4	4	4	4	3

S – N O W Y

S U N – N Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\text{diff}(i,j) \end{cases}$$

$$\text{diff}(i,j) = \begin{cases} 0, & x[i] = y[j] \\ 1, & \text{otherwise} \end{cases}$$

Space Requirements?

$n \times m$ table: $O(nm)$

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

S – N O W Y

S U N – N Y

$$E(i,j) = \min \begin{cases} \text{diff}(\quad) \\ E(i-1,j)+1 \\ E(i,j-1)+1 \end{cases}$$

Running time:

??

Spac

Can we do it in $O(\min(n,m))$ space?

Backtracking?

Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Edit Distance

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W	4	3	3	3	3	3
5	Y	5	4	4	4	4	3

S – N O W Y

S U N – N Y

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 \\ E(i,j-1)+1 \end{cases}$$

diff(

Spac

Running time:

$$O(nm(n+m))$$

Each iteration of algorithm

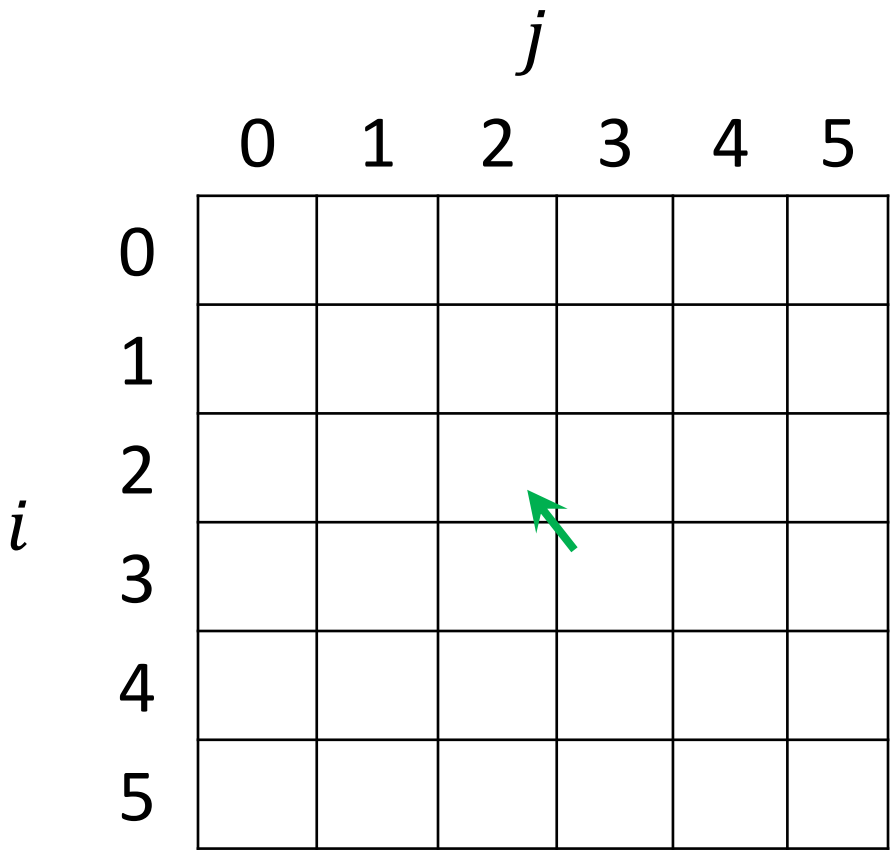
Backtracking path size

Can we do it in $O(\min(n,m))$ space?

Backtracking?

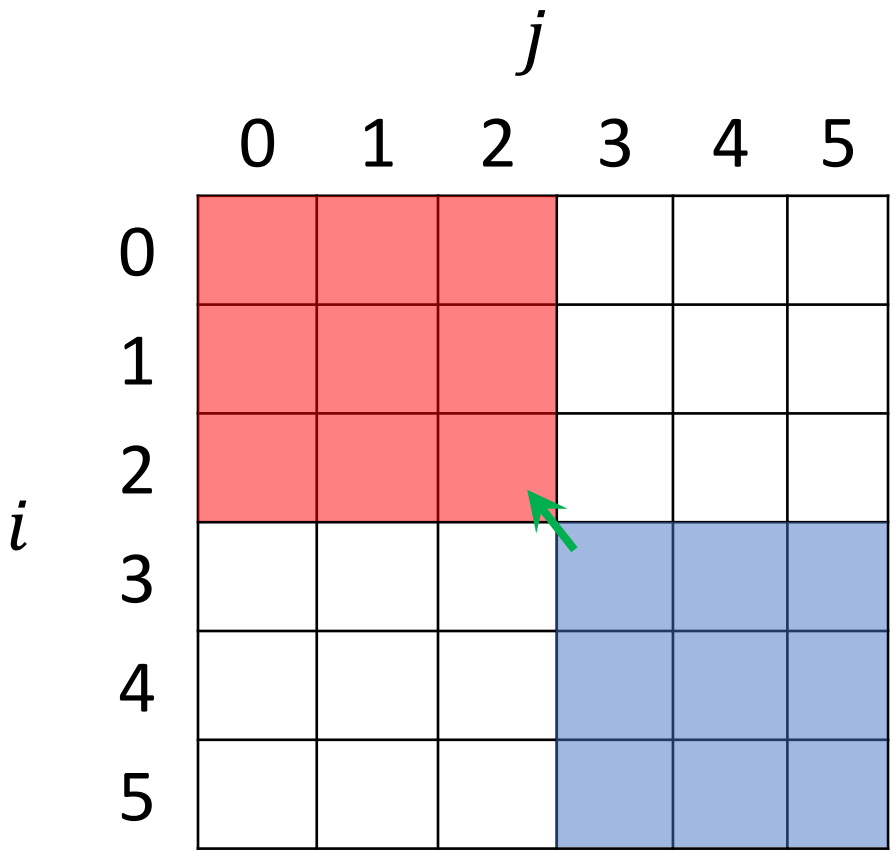
Repeatedly run algorithm up to row we are on, backtracking one step at a time.

Hirschberg's Algorithm



What if I knew for a fact that the optimal alignment went through **here**?

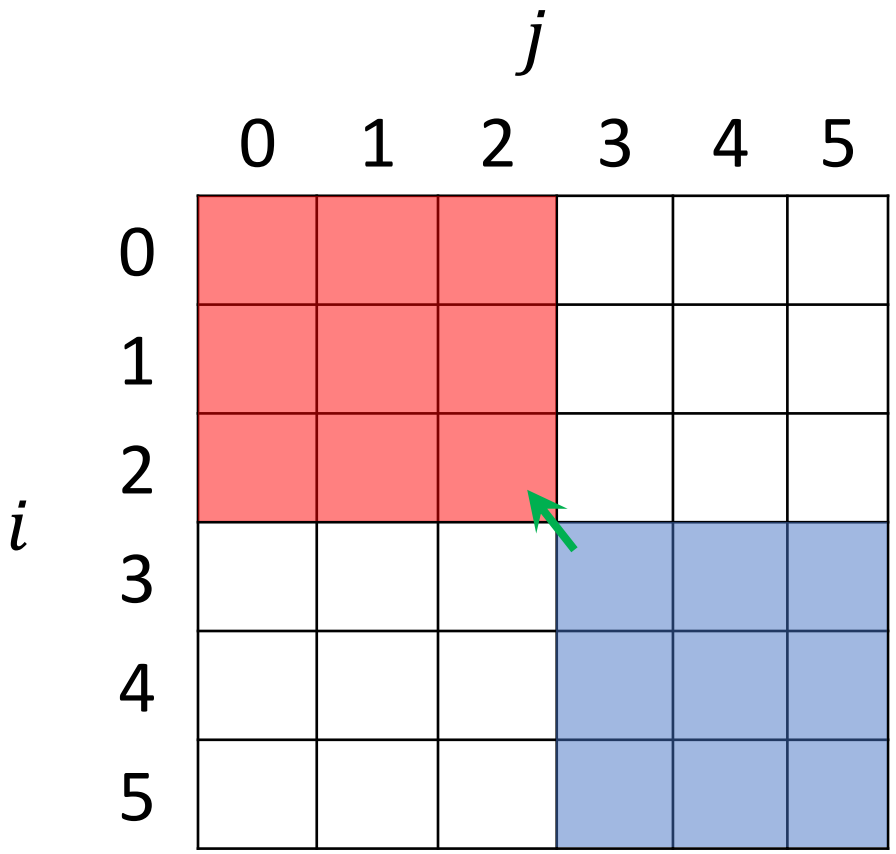
Hirschberg's Algorithm



What if I knew for a fact that the optimal alignment went through **here**?

Optimal Alignment = **Optimal alignment from start to there** + **Optimal alignment from there to end**

Hirschberg's Algorithm



What if I knew for a fact that the optimal alignment went through **here**?

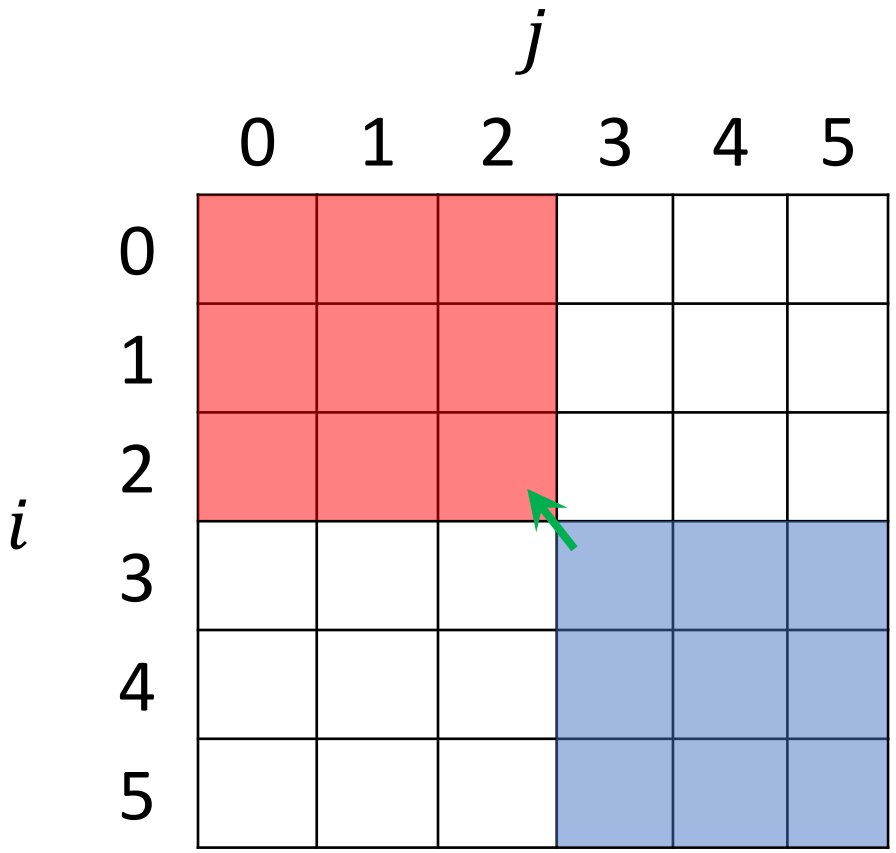
Optimal Alignment = **Optimal alignment from start to there** + **Optimal alignment from there to end**

Optimal alignment of:
$i = 0 \rightarrow 2$
and
$j = 0 \rightarrow 2$

Optimal alignment of:
$i = 3 \rightarrow 5$
and
$j = 3 \rightarrow 5$

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

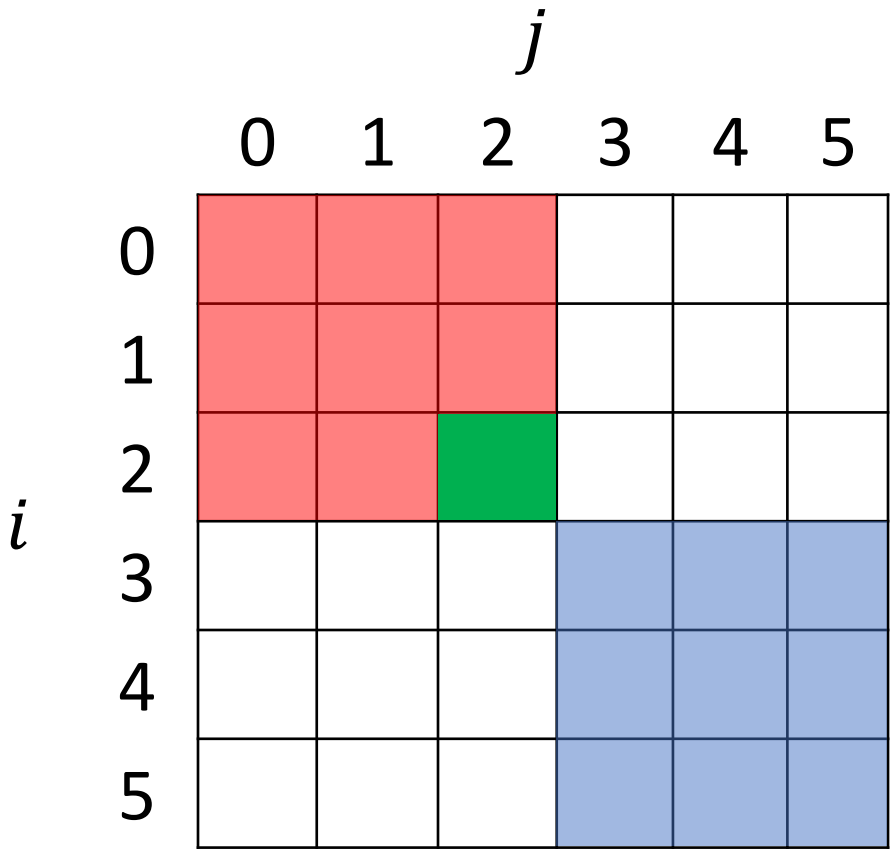


What if I knew for a fact that the optimal alignment went through **here**?

Optimal Alignment = **Optimal alignment from start to there** + **Optimal alignment from there to end**

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

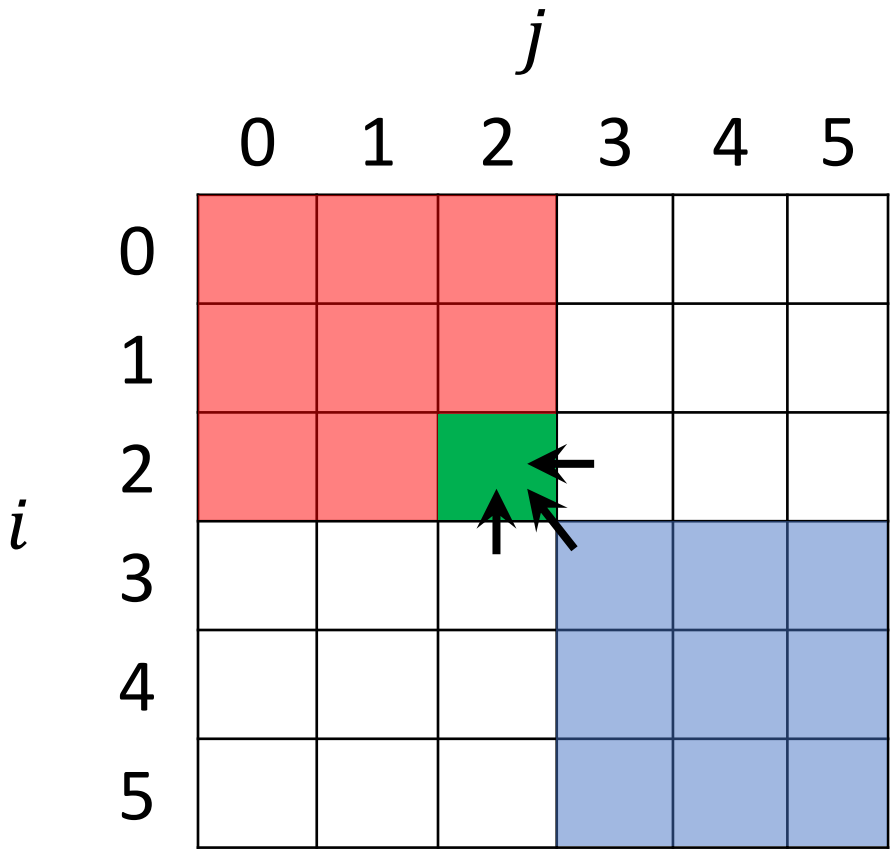


What if I knew for a fact that the optimal alignment went through **here**?

Optimal Alignment = **Optimal alignment from start to there** + **Optimal alignment from there to end**

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

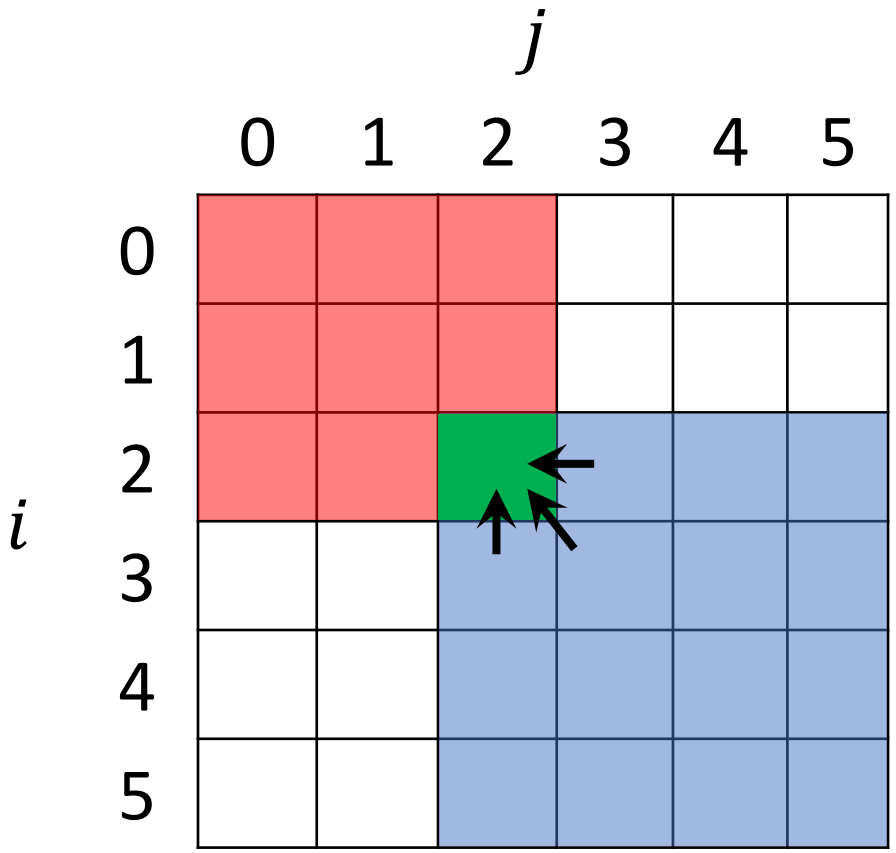


What if I knew for a fact that the optimal alignment went through **here**?

Optimal Alignment = **Optimal alignment from start to there** + **Optimal alignment from there to end**

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.



What if I knew for a fact that the optimal alignment went through **here**?

Optimal Alignment = **Optimal alignment from start to there** + **Optimal alignment from there to end**

Optimal alignment of:
$i = 0 \rightarrow 2$
and
$j = 0 \rightarrow 2$

Optimal alignment of:
$i = 2 \rightarrow 5$
and
$j = 2 \rightarrow 5$

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0							
1	S						
2	N						
3	O						
4	W						
5	Y						

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0							
1	S						
2	N						
3	O						
4	W						
5	Y						

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).

What does this value mean?

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

What does this value mean?

It is the cheapest way to align SNO with SU.

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

What does this value mean?

It is the cheapest way to align SNO with SUNN.

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).

What does this value mean?

This row is the cheapest way to align SNO with every prefix of SUNNY

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

What does this value mean?

This row is the cheapest way to align SNO with every prefix of SUNNY

We now need to align WY with every suffix of SUNNY.

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

What does this value mean?

This row is the cheapest way to align SNO with every prefix of SUNNY

We now need to align WY with every suffix of SUNNY.

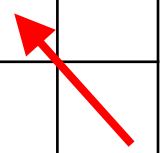
Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						



	j	0	1	2	3	4	5
i			Y	N	N	U	S
0		0	1	2	3	4	5
1	Y	1	0	1	2	3	4
2	W	2	1	1	2	3	4

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

What is happening here?

	j	0	1	2	3	4	5
i			Y	N	N	U	S
0		0	1	2	3	4	5
1	Y	1	0	1	2	3	4
2	W	2	1	1	2	3	4

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

What is happening here?

We aligned SNO with SUN.

i	j	0	1	2	3	4	5
			Y	N	N	U	S
0		0	1	2	3	4	5
1	Y	1	0	1	2	3	4
2	W	2	1	1	2	3	4

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

i		j	0	1	2	3	4	5
				S	U	N	N	Y
0			0	1	2	3	4	5
1	S		1	0	1	2	3	4
2	N		2	1	1	1	2	3
3	O		3	2	2	2	2	3
4	W							
5	Y							

What is happening here?

We aligned SNO with SUN.

What does this correspond to in R[]?

i		j	0	1	2	3	4	5
				Y	N	N	U	S
0			0	1	2	3	4	5
1	Y		1	0	1	2	3	4
2	W		2	1	1	2	3	4

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

What is happening here?

We aligned SNO with SUN.

What does this correspond to in R[]?

Aligning WY with NY.

i	j	0	1	2	3	4	5
			Y	N	N	U	S
0		0	1	2	3	4	5
1	Y	1	0	1	2	3	4
2	W	2	1	1	2	3	4

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

What is happening here?

We aligned SNO with SUN.

Cost?

What does this correspond to in R[]?

Aligning WY with NY.

i	j	0	1	2	3	4	5
			Y	N	N	U	S
0		0	1	2	3	4	5
1	Y	1	0	1	2	3	4
2	W	2	1	1	2	3	4

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

What is happening here?

We aligned SNO with SUN.

Cost? 3

i	j	0	1	2	3	4	5
			Y	N	N	U	S
0		0	1	2	3	4	5
1	Y	1	0	1	2	3	4
2	W	2	1	1	2	3	4

What does this correspond to in R[]?

Aligning WY with NY.

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

i		j	0	1	2	3	4	5
				S	U	N	N	Y
0			0	1	2	3	4	5
1	S		1	0	1	2	3	4
2	N		2	1	1	1	2	3
3	O		3	2	2	2	2	3
4	W							
5	Y							

What is happening here?

We aligned SNO with S.

Cost? 5

i		j	0	1	2	3	4	5
				Y	N	N	U	S
0			0	1	2	3	4	5
1	Y		1	0	1	2	3	4
2	W		2	1	1	2	3	4

What does this correspond to in R[]?

Aligning WY with UNNY.

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

i		j	0	1	2	3	4	5
				S	U	N	N	Y
0			0	1	2	3	4	5
1	S		1	0	1	2	3	4
2	N		2	1	1	1	2	3
3	O		3	2	2	2	2	3
4	W							
5	Y							

What is happening here?

We aligned SNO with -.

Cost? 7

What does this correspond to in R[]?

Aligning WY with SUNNY.

i		j	0	1	2	3	4	5
				Y	N	N	U	S
0			0	1	2	3	4	5
1	Y		1	0	1	2	3	4
2	W		2	1	1	2	3	4

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

i		j	0	1	2	3	4	5
				S	U	N	N	Y
0			0	1	2	3	4	5
1	S		1	0	1	2	3	4
2	N		2	1	1	1	2	3
3	O		3	2	2	2	2	3
4	W							
5	Y							

So, we know that we are splitting SNOWY as SNO|WY. What part of SUNNY should we align to SNO and what part to WY?

i		j	0	1	2	3	4	5
				Y	N	N	U	S
0			0	1	2	3	4	5
1	Y		1	0	1	2	3	4
2	W		2	1	1	2	3	4

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

	j	0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

So, we know that we are splitting SNOWY as SNO|WY. What part of SUNNY should we align to SNO and what part to WY?

	j	0	1	2	3	4	5
i			Y	N	N	U	S
0		0	1	2	3	4	5
1	Y	1	0	1	2	3	4
2	W	2	1	1	2	3	4

$F[j] + R[j] =$	7	5	4	3	3	5

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate row i as normal ($F[]$).
3. Align remainder of string 1 with string 2 (both reversed) and reverse final row ($R[]$).

j		0	1	2	3	4	5
i			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

So, we know that we are splitting SNOWY as SNO|WY. What part of SUNNY should we align to SNO and what part to WY?

j		0	1	2	3	4	5
i			Y	N	N	U	S
0		0	1	2	3	4	5
1	Y	1	0	1	2	3	4
2	W	2	1	1	2	3	4

$F[j] + R[j] =$	7	5	4	3	3	5

Hirschberg's Algorithm

Split point: A cell in the middle(ish) row that lies on at least one optimal path.

How to find the split point:

1. Identify the middle(ish) row i (string 1).
2. Calculate
3. Align rem 2 (both reversed
4. Calculate
5. Split point $= [i][j]$, where j is the j value with min $F[j] + R[j]$ value.

Complexity:
$\boldsymbol{O(nm)}$ **time**
$\boldsymbol{O(\min(n,m))}$ **space**

i	j	0	1	2	3	4	5
			S	U	N	N	Y
0		0	1	2	3	4	5
1	S	1	0	1	2	3	4
2	N	2	1	1	1	2	3
3	O	3	2	2	2	2	3
4	W						
5	Y						

$F[j] + R[j] =$	7	5	4	3	3	5

i	j	0	1	2	3	4	5
			Y	N	N	U	S
0		0	1	2	3	4	5
1	Y	1	0	1	2	3	4
2	W	2	1	1	2	3	4

Hirschberg’s Algorithm

SNOWY SUNNY

Hirschberg’s Algorithm

1. Split string 1 in half(ish).

SNO|WY SUNNY

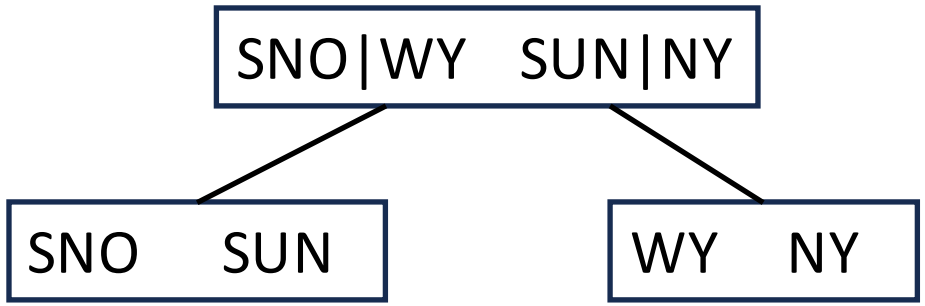
Hirschberg’s Algorithm

1. Split string 1 in half(ish).
2. Find split point in string 2.

SNO|WY SUN|NY

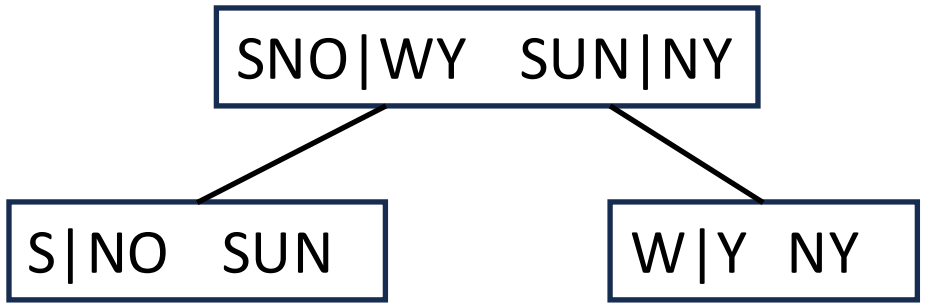
Hirschberg's Algorithm

1. Split string 1 in half(ish).
2. Find split point in string 2.
3. Recursively find the optimal alignment in the two resulting subproblems,…



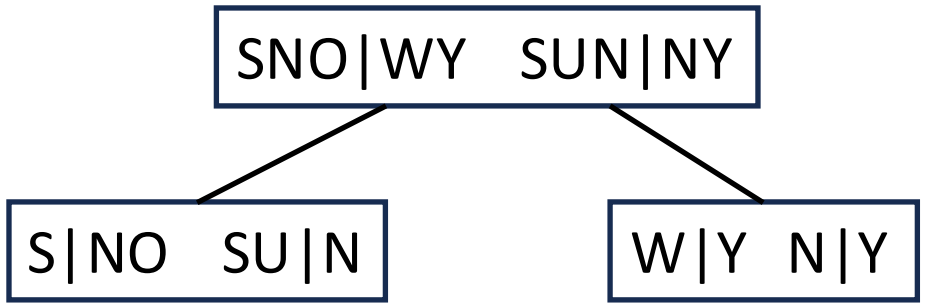
Hirschberg's Algorithm

1. Split string 1 in half(ish).
2. Find split point in string 2.
3. Recursively find the optimal alignment in the two resulting subproblems,…



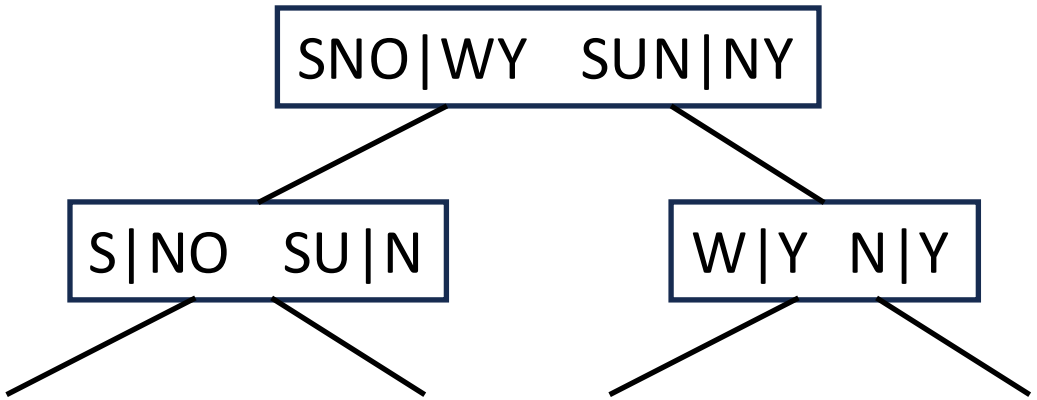
Hirschberg's Algorithm

1. Split string 1 in half(ish).
2. Find split point in string 2.
3. Recursively find the optimal alignment in the two resulting subproblems,…



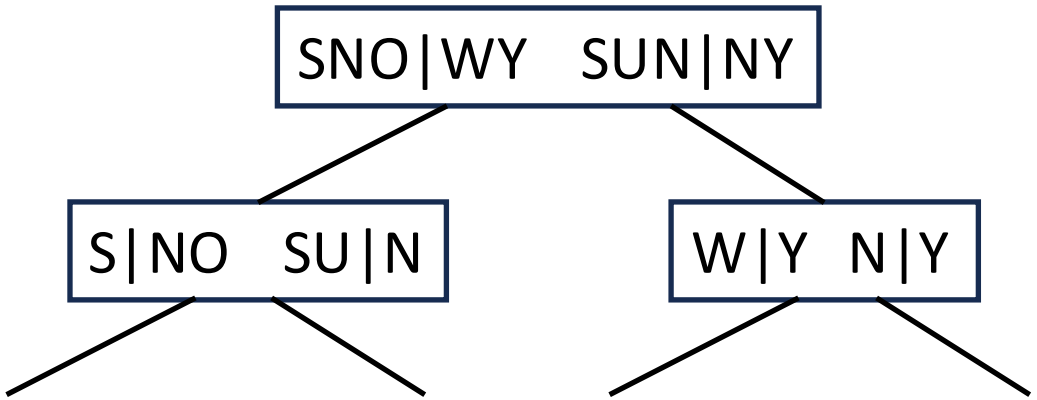
Hirschberg's Algorithm

1. Split string 1 in half(ish).
2. Find split point in string 2.
3. Recursively find the optimal alignment in the two resulting subproblems,…



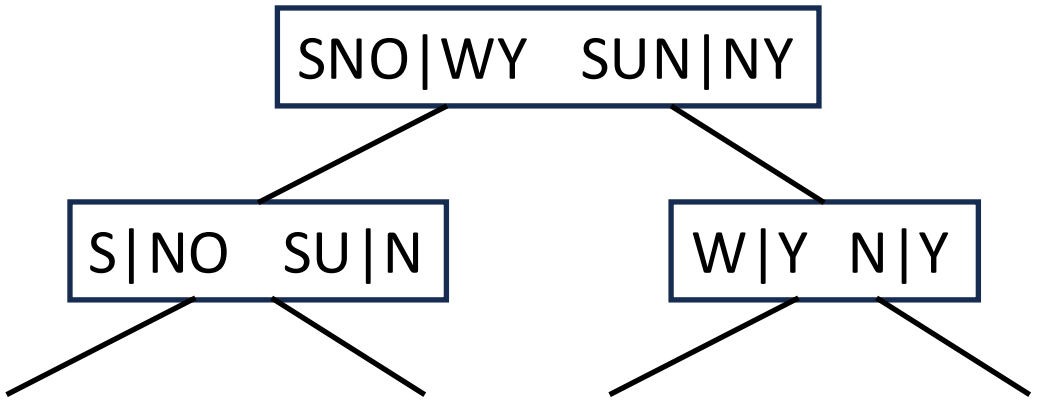
Hirschberg's Algorithm

1. Split string 1 in half(ish).
2. Find split point in string 2.
3. Recursively find the optimal alignment in the two resulting subproblems, unless string 1 is single character, then find optimal alignment manually.



Hirschberg's Algorithm

1. Split string 1 in half(ish).
2. Find split point in string 2.
3. Recursively find the optimal alignment in the two resulting subproblems, unless string 1 is single character, then find optimal alignment manually.
4. Concatenate alignments to obtain optimal alignment of input string.



Hirschberg's Algorithm

Complexity:
$\boldsymbol{O(nm)}$ **time**
$\boldsymbol{O(\min(n, m))}$ **space**

1. Split string 1 in half(ish).
2. Find split point in string 2.
3. Recursively find the optimal alignment in the two resulting subproblems, unless string 1 is single character, then find optimal alignment manually.
4. Concatenate alignments to obtain optimal alignment of input string.

