EDIT DISTANCE

D[i, j]

for strings S_1[1 … i] and S_2[1 … j]

transforms S_1 into S_2 using fewest edit operations

EDIT OPERATIONS

I Insertion
D Deletion
R Replacement
M Match

```
    R I M D M D M M I
S1  v   i n t n e r
S2  w r i   t   e r s
```

TRANSCRIPTS & ALIGNMENTS

Transcript

```
R I M D M D M M I
```

Alignment

```
S1  v   i n t n e r
S2  w r i   t   e r s
```

these are equivalent mathematically but one concerns process and the other pattern

DYNAMIC PROGRAMMING

Recurrence

D[i, 0] = i
D[0, j] = j

D[i, j] = Min[
D[i - 1, j] + 1,
D[i, j - 1] + 1,
D[i - 1, j - 1] + t[i, j]
]

where t[i, j] =
0 if $S_1[i] = S_2[j]$,
1 if $S_1[i] \neq S_2[j]$

DYNAMIC PROGRAMMING (2)

Tabulation

		w	r	i	t	e	r	s
	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
v 1	1	1	2	3	4	5	6	7
i 2	2	2	2	2	3	4	5	6
n 3	3	3	3	3	3	4	5	6
t 4	4	4	4	4	3	4	5	6
n 5	5	5	5	5	4	4	5	6
e 6	6	6	6	6	5	4	5	6
r 7	7	7	6	7	6	5	4	5

DYNAMIC PROGRAMMING (3)

Traceback

[i - 1, j]:
D[i, j] = D[i - 1, j] + 1
vertical

[i, j - 1]:
D[i, j] = D[i, j - 1] + 1
horizontal

[i - 1, j - 1]:
D[i - 1, j - 1] + t[i, j]
diagonal

D(i, j)			w	r	i	t	e	r	s
		0	1	2	3	4	5	6	7
	0	0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
v	1	↑ 1	↖ 1	↖ ← 2	↖ ← 3	↖ ← 4	↖ ← 5	↖ ← 6	↖ ← 7
i	2	↑ 2	↖ ↑ 2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
n	3	↑ 3	↖ ↑ 3	↖ ↑ 3	↖ ↑ 3	↖ 3	↖ ← 4	↖ ← 5	↖ ← 6
t	4	↑ 4	↖ ↑ 4	↖ ↑ 4	↖ ↑ 4	↖ 3	↖ ← 4	↖ ← 5	↖ ← 6
n	5	↑ 5	↖ ↑ 5	↖ ↑ 5	↖ ↑ 5	↑ 4	↖ 4	↖ ← 5	↖ ← 6
e	6	↑ 6	↖ ↑ 6	↖ ↑ 6	↖ ↑ 6	↑ 5	↖ 4	↖ ← 5	↖ ← 6
r	7	↑ 7	↖ ↑ 7	↖ 6	↖ ← ↑ 7	↑ 6	↑ 5	↖ 4	← 5

S_1 v - i n t n e r -
S_2 w r i - t - e r s

S_1 _ v i n t n e r -
S_2 w r i - t - e r s

S_1 v i n t n e r -
S_2 w r i t - e r s