Dynamic programming: Edit Distance

Edit Distance

- Given two strings s and t, the edit distance between s and t is the minimum number of editing operations needed to turn s into t

- Editing operations
  - Insertion
  - Deletion
  - Substitution

Example

```
s:  I N T E * N T I O N
    | | | | | | | | | |
t:  * E X E C U T I O N
    d s s   i s
```

- Distance: 5 (assuming unit cost for each operation)

Application: Computational Biology

- Given a sequence of bases

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC
```

- An alignment:

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

Application: NLP

Evaluating Machine Translation and speech recognition

```
R Spokesman confirms     senior government adviser was shot
H Spokesman said     the senior            adviser was shot dead
                S     I              D                       I
```

Optimal substructure?

- Q: if this is an optimal alignment

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```

will this be optimal for sure?

```
I N T E * N T I O
| | | | | | | | |
* E X E C U T I O
```

$$s = s_1\ s_2\ \ldots\ s_{m;}\quad t = t_1\ t_2\ \ldots\ t_n$$

- We can create sub-problems by considering the prefixes of s and t

D(i,j) = the edit distance between $s_1 s_2 \ldots s_i$ and $t_1 t_2 \ldots t_j$

Q: To figure out D(i,j), what are the possible choices we can make regarding the last positions i and j?

A: Three possibilities:

Align i with j

$$\begin{array}{ccccc} s_1 & s_2 \ldots & s_{i-1} & s_i \\ & & & | \\ t_1 & t_2 \ldots & t_{j-1} & t_j \end{array}$$

Align i with *

$$\begin{array}{cccccc} s_1 & s_2 \ldots\ldots & & s_{i-1} & & s_i \\ & & & & & | \\ t_1 & t_2 \ldots & t_{j-1} & & t_j & * \end{array}$$

Align j with *

$$\begin{array}{cccccc} s_1 & s_2 \ldots & s_{i-1} & s_i & * \\ & & & & | \\ t_1 & t_2 \ldots & t_{j-1} & & t_j \end{array}$$

- D(i,j): the minimum of the three possible choices:

Align i with j

$$\begin{array}{cccc} s_1\, s_2 \ldots & s_{i-1} & s_i & \\ t_1\, t_2 \ldots & t_{j-1} & t_j & \end{array}$$

Choice 1:

If $s_i = t_j$:
$D(i, j) = D(i-1, j-1)$

Otherwise:
$D(i, j) = D(i-1, j-1) + 1$

Align i with *

$$\begin{array}{cccc} s_1\, s_2 \ldots\ldots & s_{i-1} & & s_i \\ t_1\, t_2 \ldots & t_{j-1} & t_j & * \end{array}$$

Choice 2:

$D(i, j) = D(i-1, j) + 1$

Align j with *

$$\begin{array}{cccc} s_1\, s_2 \ldots & s_{i-1} & s_i & * \\ t_1\, t_2 \ldots & t_{j-1} & & t_j \end{array}$$

Choice 2:

$D(i, j) = D(i, j-1) + 1$

Recurrence relation for D(i,j)

For $i, j \geq 1$

$$D(i,j) = \min \begin{cases} D(i-1, j) + 1 & \boxed{\text{deletion}} \\ D(i, j-1) + 1 & \boxed{\text{insertion}} \\ D(i-1, j-1) + \text{diff}(s_i, t_j) & \boxed{\text{align i with j}} \end{cases}$$

$$\text{diff}(a, b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{if } a \neq b \end{cases}$$

Base case? $D(0,0) = 0$

any others?

Edit Distance

```
For i=0 to m: D(i,0) = i
For j=1 to n: D(0,j) = j
For each  i = 1…m
    For each  j = 1…n
```

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \mathrm{diff}(s_i,\ t_j) \end{cases}$$

```
Return D(m,n)
```

The Edit Distance Table

9	N										
8	O										
7	I										
6	T										
5	N										
4	E										
3	T										
2	N										
1	I										
0	#										
j=		#	E	X	E	C	U	T	I	O	N
	i=	0	1	2	3	4	5	6	7	8	9

Computing alignments

- Getting the edit distance isn't sufficient
  - We often need to **align** each character of the two strings to each other
- We do this by keeping a "backtrace"
- Every time we enter a cell, remember where we came from
- When we reach the end,
  - Trace back the path from the upper right corner to read off the alignment

Adding Backtrace to Minimum Edit Distance

- Base conditions:

  `D(i,0) = i` `D(0,j) = j`

Termination:

`return D(m,n)`

- Recurrence Relation:

  For each $i = 1\ldots M$

  For each $j = 1\ldots N$

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \boxed{\text{deletion}} \\ D(i,j-1) + 1 & \boxed{\text{insertion}} \\ D(i-1,j-1) + \mathrm{diff}(s_i, t_j) & \boxed{\text{align}} \end{cases}$$

$$ptr(i,j) = \begin{cases} \text{LEFT} & \boxed{\text{insertion}} \\ \text{DOWN} & \boxed{\text{deletion}} \\ \text{DIAG} & \boxed{\text{align}} \end{cases}$$

Result of Backtrace

- Two strings and their **alignment**:

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```

Performance

- Time:

  $O(nm)$

- Space:

  $O(nm)$

- Backtrace

  $O(n+m)$