Announcements

- HW is up
Edit Distance

The edit distance between two words is the minimum number of letter insertions, letter deletions, and letter substitutions required to transform one word into another.

Ex: Food to money: 4

Food → Mood → Mond → Money

edit distance ≤ 4 edit distance ≥ 3
Better display:

\[ \text{FOOD} \ 
\downarrow \downarrow \downarrow \downarrow \\
\text{MONEY} \]

Why can't you get 3?

at least 3 different letters,
plus FOOD is shorter
Another: Algorithm to All-Trust

AL G O R I T H M
AL T R U I S T I C

1 + 1 + 1 + 1 + 1 + 1 = \sqrt{6}

so edit dist \leq 6
Recursive idea:
Suppose we remove the last column.

What do we know about rest?

ALGORITHM

AL TRUISTIC

spots could do better on substring
Lemma: If we remove last column, the remaining columns must represent the shortest edit sequence of remaining substrings.

pf: by contradiction

If substring had a better edit sequence then we could find a better edit sequence for the whole word.
So - recursive definition of my two words

Consider words $A[1..m] + B[1..n]$

- if $A[m] = B[n]$, then $A[m]B[n]$,
- if $A[m] \neq B[n]$, rest +1

What could happen in last column?

- have substitution
- have insertion
- have deletion
Formally:

$$\text{Edit}(A[1..m], B[1..n]) = \min \left\{ \begin{array}{l} \text{Edit}(A[1..m-1], B[1..n]) + 1 \\ \text{Edit}(A[1..m], B[1..n-1]) + 1 \\ \text{Edit}(A[1..m-1], B[1..n-1]) + [A[m] \neq B[n]] \end{array} \right\}$$

Base cases:

$$\text{Edit}(A[1..m], \varepsilon) = m, \quad \text{Edit}(\varepsilon, B[1..n]) = n.$$
So (recap):

\[
\text{Edit}(i, j) = \begin{cases} 
    i < j & \text{if } j = 0 \\
    j & \text{if } i = 0 \\
    \min \left\{ \begin{array}{l}
    \text{Edit}(i - 1, j) + 1, \\
    \text{Edit}(i, j - 1) + 1, \\
    \text{Edit}(i - 1, j - 1) + [A[i] \neq B[j]]
    \end{array} \right\} & \text{otherwise}
\end{cases}
\]

This gives a (nasty) recurrence:

\[
T(0, n) = O(n) \\
T(m, 0) = O(1) \\
T(m, n) = T(m-1, n) + T(m, n-1) + T(m-1, n-1) + O(1)
\]
A trick - replace \( m + n \) with a single variable, \( N = m + n \).

Then:

\[
T(m, n) = \begin{cases} 
O(1) & \text{if } n = 0 \text{ or } m = 0, \\
T(m, n - 1) + T(m - 1, n) + T(n - 1, m - 1) + O(1) & \text{otherwise.}
\end{cases}
\]

Becomes:

\[
T'(N) = \max_{n + m = N} T(n, m) = \begin{cases} 
O(1) & \text{if } N = 0, \\
2T(N - 1) + T(N - 2) + O(1) & \text{otherwise.}
\end{cases}
\]

(worse than Fibonacci !) way exponential
Smart recursion (aka memoization)

Keep 2 dimensional table \( Edit(m, n) \):

\[ Edit(i, j) = \text{the edit distance between } A[i..j] \text{ and } B[j..j] \]

\[ Edit(i, j) \]
Space? How big is table? $n \times m$

How much time per entry?

- need to check 3 other entries
- maybe add 1
- take min

$\Rightarrow O(d)$

Total: $O(mn)$
Pseudocode:

```
EDITDISTANCE(A[1..m], B[1..n]):
    for j ← 1 to n
        Edit[0, j] ← j
    for i ← 1 to m
        Edit[i, 0] ← i
        for j ← 1 to n
            if A[i] = B[j]
                Edit[i, j] ← min {Edit[i - 1, j] + 1, Edit[i, j - 1] + 1, Edit[i - 1, j - 1]}
            else
                Edit[i, j] ← min {Edit[i - 1, j] + 1, Edit[i, j - 1] + 1, Edit[i - 1, j - 1] + 1}
    return Edit[m, n]
```

\(O(nm)\)
An example - Algorithm to Altruistic

[horizontal = deletion]
[vertical = insertion]
[diagonal = substitution]

Any path from top left to bottom right represents a valid edit sequence.
In our example, there are actually 3 optimal sequences:

1. ALGORITHM
   ALTRUISTIC
2. ALGORITHM
   ALTRUISTIC
3. ALGORITHM
   ALTRUISTIC