Q: Define similarity?

Example:
- covid
- cove
- novel

Example:
- vaccine
- vac
- mazine
- mag

def: \( x, y \in \mathbb{Z} \) are alphabet \( \Sigma \). The "edit distance" \( \text{dist}(x, y) \), is min
- number of - substitutions
- insertions
- deletions

required to transform \( x \) into \( y \)

Let: \( \text{dist}(x, y) = \text{dist}(y, x) \)
An alignment between $x, y \in \Sigma^*$ can be defined as a function $\text{dist}(x, y) = \min_{\pi(x, y)} \sum_{(i, j) \in \pi} c_{i,j}$, where $c_{i,j}$ is a cost matrix.

We say $x_i$ is matched to $y_j$ in $A$. The score of an alignment is

$$|\{ (i, j) \in A : x_i \neq y_j \}| + |\{ i : x_i \text{ unmatched in } A \}| + |\{ j : y_j \text{ unmatched in } A \}|$$

\[eg.\] 

\[\text{score: } 11 + 1 + 1 + 1 = 4\]
Q: compute edit distance? ^
  efficiently?

Q: what are the substrings?

idea, compute edit distance between all substrings of x
  all substrings of y

continued
\[ \text{len} : \text{alignments of } x, y \]

\[ \text{if } x_n, y_n \text{ both matched in } A \]

\[ \Rightarrow (x_n, y_n) \in A \]

\[ k = \text{yn matched to } y_k \Rightarrow (n, k) \in A \Rightarrow n \leq k \Rightarrow k \leq m \]

\[ \text{yn matched to } x \in (\text{len}) \setminus \{x_n\} \quad \text{len} \quad k \leq m \]

\[ \text{cars} : \{ \text{alignments of } x, y \} = \]

\[ \bigcup \left\{ \text{alignments of } x, y, \text{ where } x_n \text{ matched to } y_m \right\} = \left\{ A : (x_n, y_m) \in \text{align of } x_n, y_m \right\} \]

\[ \bigcup \left\{ \text{alignments of } x, y, \text{ where } x_n \text{ unmatched } \right\} = \left\{ A : A \text{ align of } x, y \right\} \]

\[ \bigcup \left\{ \text{alignments of } x, y, \text{ where } y_m \text{ unmatched } \right\} = \left\{ A : A \text{ align of } x, y \right\} \]

\[ \text{match} : \{ \text{cars} \} \text{ dynamic programming} \]

\[ \text{cars} : \text{dist}(x, y) = \min \bigg( \text{dist}(x_n, y_m), \text{dist}(x, y) + 1, \text{dist}(x, y) + 1 \bigg) \]

\[ \text{dist}(x_n, y_m) = \Delta \iff x_n, y_m \text{ matched}, \quad \text{is at min} \]
Given:

\[ d^{\text{init}}(x, y) = |x| \]

\[ d^{\text{init}}(x, y) = |y| \]

Prop: \( d(x, y) \) computable in \( O(nm) \) time

Proof:

1. for \( 0 \leq i < n \), \( d[i][0] = i \)
2. for \( 0 \leq j < n \), \( d[0][j] = j \)
3. for \( 0 \leq i < n \), \( 0 \leq j < n \)
   \[ d[i][j] \leftarrow \min \left( d[i-1][j-1] + 1, d[i-1][j] + 1, d[i][j-1] + 1 \right) \]
4. return \( d[n][m] \)

Correctness:
\[ d(x, y) = d[i][j] \]

Complexity: \( O(nm) \)

Question: Is this good?
- Yes: poly space
- No: \( n \times n \) large in modern contexts, time is cheaper than space
\[ d(i, j) = \text{dist}(x_i, y_j) \]

**idea:** to compute with column, only need
- \((i-1)\)st column
- Oth one

\[ \text{dist}(x_i, y_j) \text{ computed in } O(n m) \text{ time} \]

\[ O(m) \text{ space} \]

**Wt. also:**
1. \(0 \leq j \leq m \land d[\text{prev}][j] = \)
2. \(1 \leq i \leq \)
   - \((a) d[\text{rev}][0] = i\)
   - \(1 \leq j \leq m \land d[\text{rev}][j] = \min \)
   - \(d[\text{prev}][i] = d[\text{rev}][j] \)
   - \(d[\text{rev}][j] = 1 \)
   - \(d[\text{rev}][j-1] + 1 \)
   - \(d[\text{rev}][j] = 1 \)
   - \(d[\text{rev}][j-1] + 1 \)
Q: Can you explain the alignment?

Prop: Given \( \{ \text{dist}(x_i, y_j) \} \) for \( 0 \leq i, j \leq n \), can we optimally align in \( O(nm) \) time?

Sketch:

\[ \text{dist}(x, y) \] in \( O(nm) \) space

Idea: Divide and conquer?

Vacation: best case?

Q: Can you describe the alignment?

Prop: \( 1 \leq i \leq n \)

\[ \{ \text{align}(x_i, y_j) \} = \bigcup \{ A \in A > \text{align} x_i, y_j \} \]

\( A \in A > \text{align} \)

\( x > i, y > j \)

0: \( 1 \leq i \leq n \)

\[ \text{dist}(x, y) = \min \left( \text{dist}(x_i, y_j) \right) + \text{dist}(x_i, y_j) \]
Prop.

\[ \text{dist}(x, y) = \min \{ \text{dist}(x', y) : x' \in A \} \]

Lemma.

Let \( a = \arg \min_j \text{dist}(x_i, y_j) \) be computed in \( \text{dist}(x_i, y_j) \).

\[ (x, y) = (x_{a}, y_{j_{a}}) \]

\[ \text{time} = \text{time} + \text{time} \text{dist}(x_{a}, y_{j_{a}}) \]

T(x, y) = T(x_{a}, y_{j_{a}}) + T(x_{a}, y_{j_{a}})
Today: dynamic program

- Edit dist
  - O(nm) time
  - O(nm) space
- Value
- Alignment

Next lecture: dynamic program

Logistics:
- Get 1 due Fri + 1 grage ≤ 3 pp
- Lecture notes resource next week