CS473 Algorithms: Lecture 5 (2022-02-01)

logistics:
- pset 1 due F17 ⟵ groups of ≤ 3 students!
- lectures online, in person next week

last lecture: dynamic programming
- abstract
  - trees vs DAG
  - memoization vs iterative
- knapsack
  - parameterize subproblems w/ new variable
  - pseudo poly time algo

today: dynamic programming

Q: origin of omicron variant?

coronavirus genomes are RNA
= strings over A, G, C, U

idea: place omicron close to most similar variant

Q: define similarity?

eg:
covid
covi**l**
co**ve**l ⟵ 3 changes
novel

eg:
vaccine
vac**z**ine
maczine
ma**g**zine
maga zine

def: $x, y \in \Sigma^*$ over alphabet Σ. The edit distance, $dist(x,y)$, is min number of
- substitutions
- insertions
- deletions

required to transform x into y

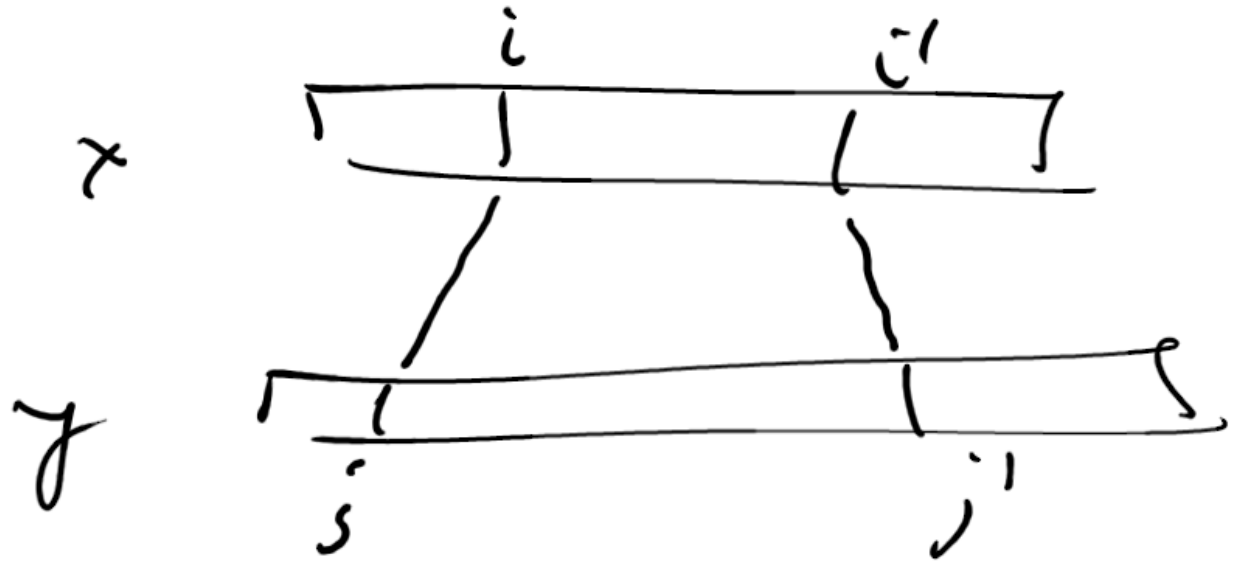
lem: $dist(x,y) = dist(y,x)$

def: $x, y \in \Sigma^*$, $n = |x|$, $m = |y|$. An alignment between x, y is a set $A \subseteq [n] \times [m]$

s.t. for $(i,j) \neq (i',j')$ either

$$i < i', \; j < j' \quad \text{or} \quad i > i', \; j > j'$$

eg:



we say x_i is matched to y_j in A

the cost of 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: vaccine / magazine — insertion, 3 substitutions

rmk: can generalize cost function

fact: $dist(x,y)$ = min cost align between x, y

Q: compute edit distance? efficiently?

Q: what are the subproblems?

vaccine

magazine

idea: compute edit distance between

all substrings of x

all (contiguous) substrings of y

Lem: A alignment of x, y

if x_n, y_m both matched in A

$\Rightarrow (x_n, y_m) \in A$

Pf: x_n matched to $y_k \Leftrightarrow (n,k) \in A \Rightarrow m \le k \Rightarrow k = m$

y_m matched to $x_l \equiv (l,m) \in A$ $\quad l \le n \qquad k \le m$

Cor: {alignments of x, y} =

$\bigcup$ {alignments of x, y, where x_n matched to y_m} $= \{A \cup (x_n, y_m) : A$ align of $x_{<n}, y_{<m}\}$

{ ———, x_n unmatched } $= \{A : A$ align of $x_{<n}, y\}$

{ ———, y_m unmatched } $= \{A : A$ align $x, y_{<m}\}$

rmk: (non) disjoint decomposition

Cor:
$$\mathrm{dist}(x,y) = \min \begin{cases} \mathrm{dist}(x_{<n}, y_{<m}) + \mathbb{1}[x_n \neq y_m] \\ \mathrm{dist}(x_{<n}, y) + 1 \\ \mathrm{dist}(x, y_{<m}) + 1 \end{cases}$$

exists opt align of x, y w/ x_n, y_m matched, iff [the first term] is argmin lem: ε empty string

$dist(x, \varepsilon) = |x|$

$dist(\varepsilon, y) = |y|$

prop: $dist(x, y)$ computable in $O(nm)$ time [and] $O(nm)$ space

pf: algo.

$O(n)$ (1) for $0 \le i \le n$ $d[i][0] = i$

$O(m)$ (2) for $0 \le j \le m$ $d[0][j] = j$

$O(n)$ (3) for $1 \le i \le n$

$O(m)$ (a) for $1 \le j \le m$

$O(1)$ (i) $d[i][j] = \min \begin{cases} d[i-1][j-1] + \mathbb{1}[x_i \neq y_j] \\ d[i-1][j] + 1 \\ d[i][j-1] + 1 \end{cases}$

(4) return $d[n][m]$

correctness: $dist(x_{\le i}, y_{\le j}) = d[i][j]$

complexity: $O(nm)$ $\square$

algo. ______

correctness: ______

complexity: $d[\cdot][\cdot]$ is $O(nm)$ entries, each $O(1)$ size

Q: [space] complexity?

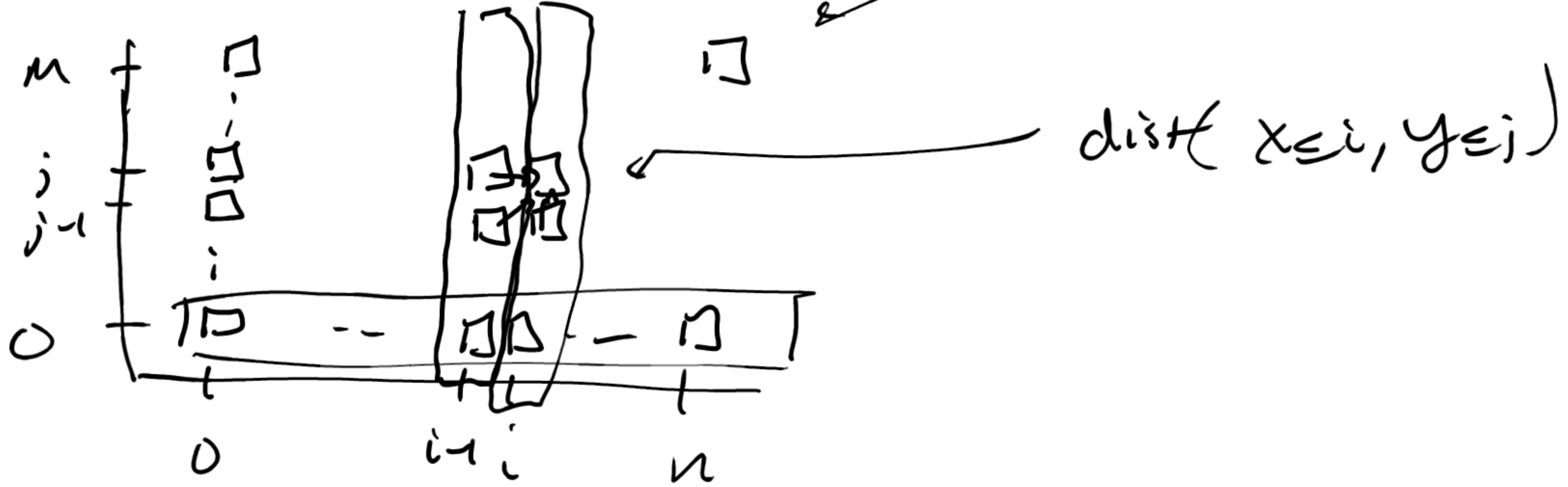
⇐ RAM

Q: is this good?

yes: poly space

no: n,m large
in modern computers, time is cheaper than space

Q: can we do better?



idea: to compute ith column, only need
- (i-1)st column
- 0th row

prop: dist(x,y) computed in $O(nm)$ time
$O(m)$ space

pf: algo:

(1) for $0 \le j \le m$ $d[prev][j] = j$

(2) for $1 \le i \le n$

(a) $d[cur][0] = i$

(b) for $1 \le j \le m$

(i) $d[cur][j] = \min \begin{cases} d[prev][j-1] + \mathbb{1}[x_i \ne y_j] \\ d[prev][j] + 1 \\ d[cur][j-1] + 1 \end{cases}$

(c) $d[prev][\cdot] \leftarrow d[cur][\cdot]$

(3) return $d[cur][m]$

correctness: clear

complexity: $O(nm)$ time

space $d[cur][\cdot]$, $d[prev][\cdot]$ $\Rightarrow O(m)$ space

∎

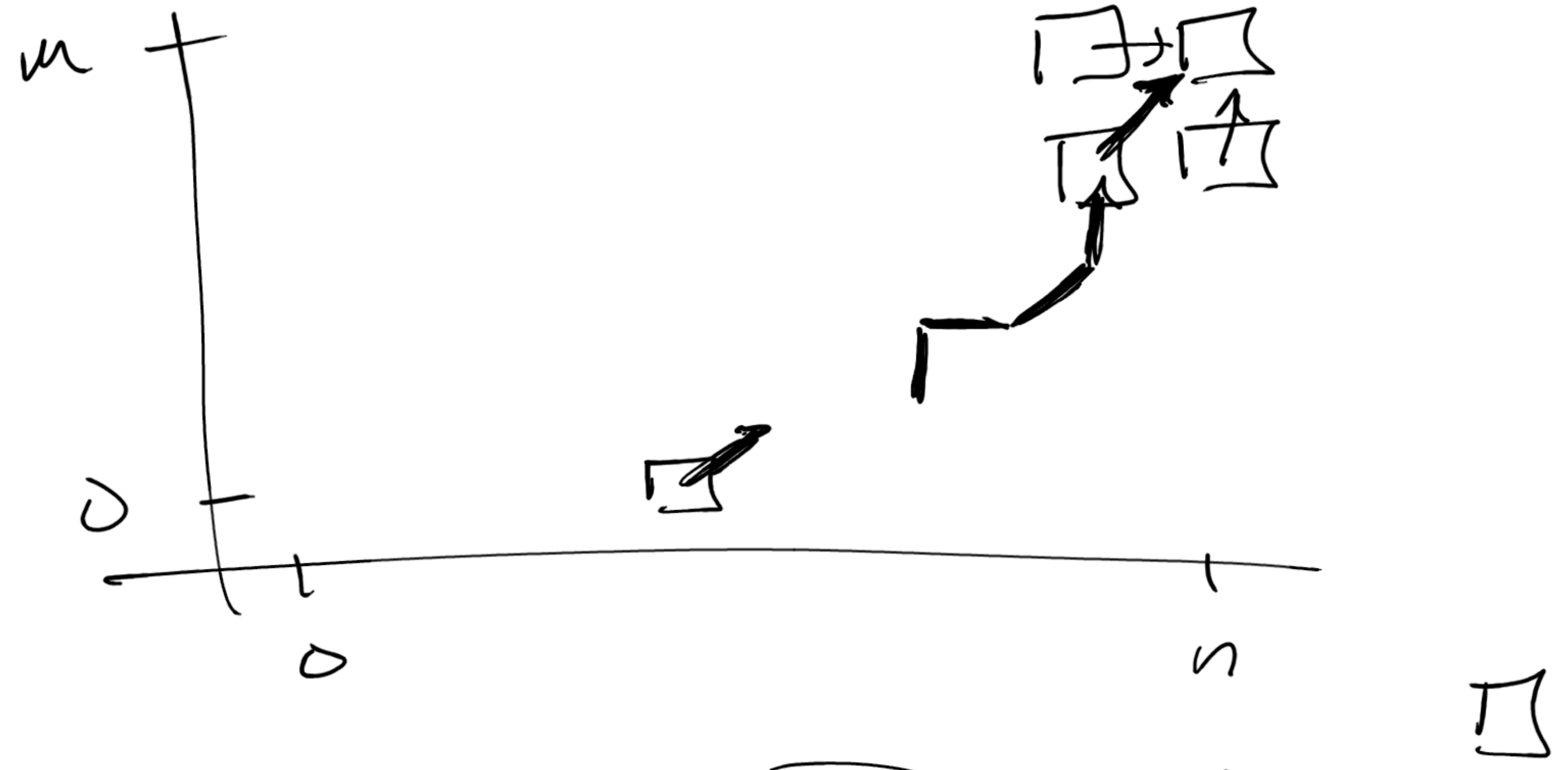
rmk: by symmetry can get $O(\min\{n,m\})$ space

Q: computing the alignment?

prop: given $\{ dist(x_{\le i}, y_{\le j}) \}_{0 \le i \le n,\ 0 \le j \le m}$

⇒ compute optimal alignment in $O(n+m)$ time

sketch:



Q: compute alignment in (small) space?

alignment(x, y) via table of size $O(nm)$

dist(x, y) in $O(n+m)$ space

idea: divide and conquer?

vaccine
massazine

idea: reuse space

~~vac mas~~ → erased

cine zine

prop: $1 \le i \le n$

$\{\text{alignments of } x, y\} = \bigcup_{0 \le j \le m} [A_\le \circ A_> : A_\le \text{ aligns } x_{\le i}, y_{\le j};\ A_> \text{ aligns } x_{>i}, y_{>j}]$

cor: $1 \le i \le n$

$dist(x, y) = \min_j (dist(x_{\le i}, y_{\le j}) + dist(x_{>i}, y_{>j}))$

prop: $\{dist(x, Y_{\le j})\}_{0 \le j \le m}$ can be computed in $O(nm)$ time, $O(m)$ space

—— $y_{>j}$ ——

def: $1 \le i \le n$

$$\mathrm{Meet}_i(x,y) := \arg\min_j \; dist(x_{\le i}, y_{\le j}) + dist(x_{>i}, y_{>j})$$

↳ computable in $O(nm)$ time, $O(m)$ space

prop: optimal alignment can be computed in $O(nm)$ time, $O(n+m)$ space

pf: algo: align-concise(x,y) =
(1) if $n=1$ return align(x,y)
(2) if $n \ne 1$ ——
(3) $j^* = \mathrm{meet}_{n/2}(x,y)$
(4) $A_{\le}$ = align-concise$(x_{\le n/2}, y_{\le j^*})$
(5) $A_{>}$ = align-concise$(x_{>n/2}, y_{>j^*})$
(6) return $A_{\le} \circ A_{>}$

correctness: clear

complexity:

space: $S(n,m) = \max \begin{cases} O(m) \\ O(n+m) \\ S(n/2, j^*) \\ S(n/2, m-j^*) \end{cases}$

$= O(n+m)$

time: $T(n,m) \le \underbrace{O(nm)}_{\alpha \cdot nm} + T(n/2, j^*) + T(n/2, m-j^*)$

guess: $T(n,m) \le \beta \cdot nm$

$\le \alpha nm + \beta \cdot \frac{n}{2} \cdot j^* + \beta \cdot \frac{n}{2}(m - j^*)$

$= (\alpha + \frac{\beta}{2}) nm$ $\quad \beta \cdot \frac{n}{2} m$

$\le \beta nm$ if $\beta \ge 2\alpha$

∎

today: dynamic programming

- edit distance
  - $O(nm)$ time $O(nm)$ space
    - value
    - alignment
  - $O(nm)$ time $O(n+m)$ space
    - value
    - alignment

next lecture: dynamic programming

logistics:
- pset 1 due F17 ← groups ≤ 3 ppl
- lectures online in person next week