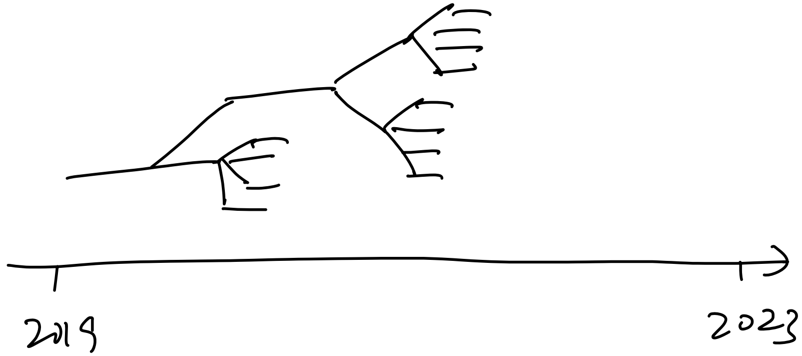
How is COVID evolving ?

String of A, G, C, U



Today: understanding similarity.

C O V I D
C O V I L
C O V E L
N O L E L

$\Rightarrow$ three changes . . .

EDIT DISTANCE

def: $x, y \in \Sigma^*$, edit distance $\mathrm{dist}(x,y)$

is min # (substitution, insertion, deletion) to change x to y.

def: $x, y \in \Sigma^*$, $n = |x|$, $m = |y|$,

__alignment__ of x, y $A \subseteq [n] \times [m]$ s.t.

$$(i, j) = (i', j') \in A$$

either $\begin{cases} i < i' \\ j < j' \end{cases}$ or $\begin{cases} i > i' \\ j > j' \end{cases}$

fact: $\mathrm{dist}(x,y) =$ min cost of align of x, y.

What are subproblems?

e.x. v a c c i n e

m a g a z i n e

idea: compute edit distance between all substring of x, all substr of y.

lemma

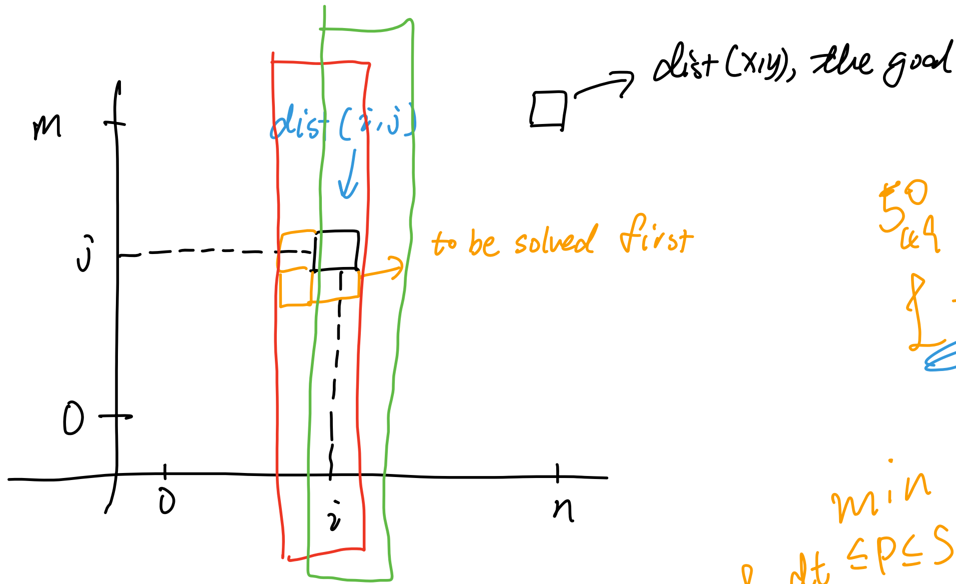
RAM usage.

prop $O(nm)$ space

complexity: $d[\cdot][\cdot]$ has $O(nm)$ space.

But, not good enough in practice. Think of n, m are huge.

Better



Note:
computing the i^{th} col, only need column $i-1$

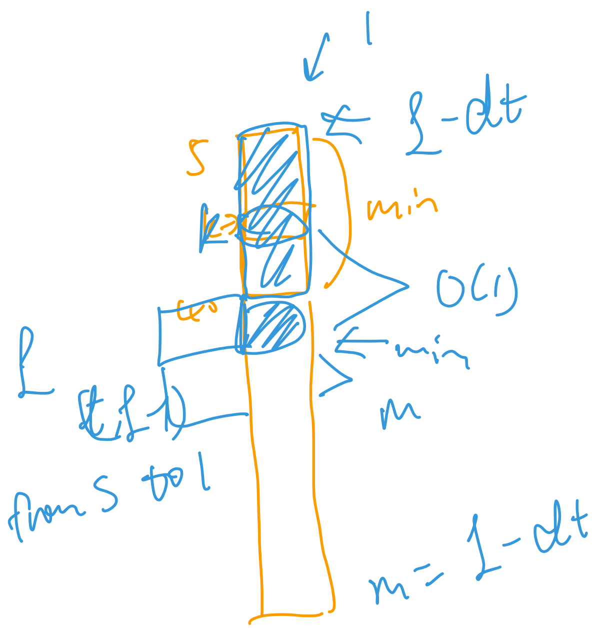
50   10

$l > dt$

min

$l - dt \le p \le s$

For l from s to l



$s - dt \le p \le s$

prop space complexity is $O(m)$

algo:
- for $0 \le j \le m$, $d[prev][j] = j$ base
- for $1 \le i \le n$,
  - $d[cur][0] = i$ ← 0^{th} row, equal to $d[i][0]$ in old algo.
  - for $1 \le j \le m$
    - $d[cur][j] = \min \begin{cases} d[prev][j-1] + 1 \\ d[prev][j] + 1 \\ d[cur][j-1] + 1 \end{cases}$
  - $d[prev][\cdot] = d[cur][\cdot]$
  - clear $d[cur][\cdot]$

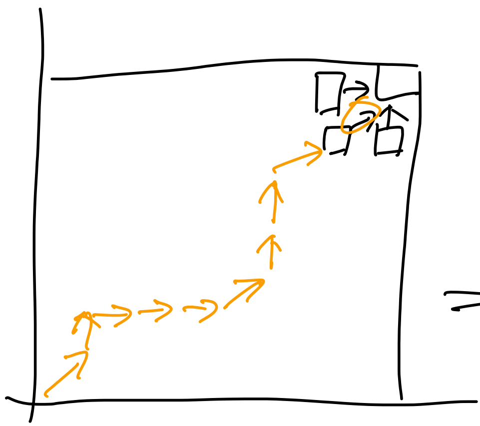
Complexity: only use $O(2 \cdot m)$ space for d.

Q: Compute alignment ?

prop given $\{dist(x_{\le i}, y_{\le j})\}_{\substack{1\le i\le n \\ 1\le j\le m}}$

compute alignment in $O(n+m)$ time.

That is, given $d[1..n][1..m]$ computed by "old" algo, this can be done in $O(n+m)$



trace the path, which has length $O(n+m)$.

$\Rightarrow$ $O(nm)$ space)

Q: compute align in small space ?

Say $O(nm)$ time, $O(m)$ space.

idea: divide and conquer.

split string <u>evenly</u>.

idea: <u>reuse</u> space

non-crossing behaviour.

prop: $1 \leq i \leq n$ fixed

$$\{\text{align}(x,y)\} = \bigcup_{0 \leq j \leq m} \left\{ A_\leq \cdot A_> : \begin{array}{l} A_\leq \text{ align } x_{\leq i} \text{ to } y_{\leq j} \\ A_> \text{ align } x_{>i} \text{ to } y_{>j} \end{array} \right\}$$

Cor

$$\text{dist}(x,y) = \min_j \left\{ \text{dist}(x_{\leq i}, y_{\leq j}) + \text{dist}(x_{>i}, y_{>j}) \right\}$$

prop

i fixed, we can compute

- $\{\text{dist}(x_{\leq i}, y_{\leq j})\}_{0 \leq j \leq m}$ in $O(n \cdot m)$ time, $O(m)$ space.
- $\{\text{dist}(x_{>i}, y_{>j})\}_{0 \leq j \leq m}$ in $O(n \cdot m)$ time, $O(m)$ space.

prop i fixed, we can compute in $O(n \cdot m)$ time, $O(m)$ space.

Prop optimal align in $O(nm)$ time, $O(m)$ space.

$O(m+n)$?

Pf :

algo align-concise(x, y)

- if $n=1$, return align(x,y)
- if $m=1$, return align(x,y)
- $j^* = \text{meet}_{i=\frac{n}{2}}(x, y)$
- $A_{\le}$ = align-concise$(x_{\le \frac{n}{2}}, y_{\le j^*})$
- $A_{>}$ = align

Complexity :

$$S(n,m) \le \underbrace{\max}_{\text{we can reuse!!!}}\Big(\overbrace{O(n+m)}^{\text{computing } j^*}, \overbrace{S(\tfrac{n}{2}, j^*), S(\tfrac{n}{2}, m-j^*)}^{\text{recurse, smaller than } O(n+m)}\Big) \le O(n+m)$$