Edit distance:

- minimum # of insertions, deletions, & substitutions to turn one string into another.

FOOD → MOOD → MOND

→ MONED → MONEY

⇒ edit distance ≤ 4

(Levenshtein or

Ulam distance)

Given two strings

$A[1..m]$ & $B[1..n]$.

Goal: compute their edit distance

Want to figure out what happened to each character of A / where did B's chars come from.

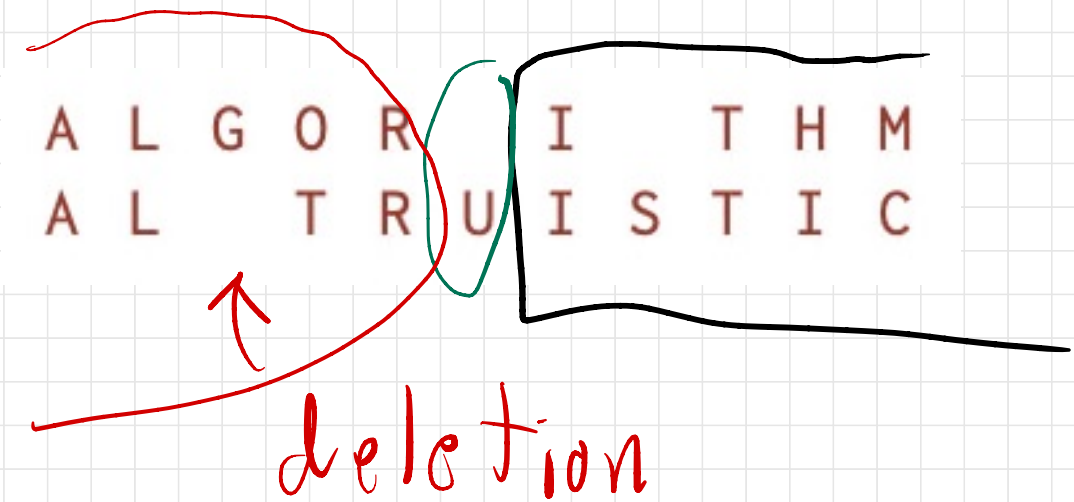
FOOD

MONEY

insertion

substitution

free



For each choice of the kth edit, recursively try to minimize # of edits between the remaining prefixes.

Edit(i, j): Edit distance between $A[1..i]$ + $B[1..j]$.

Ultimate goal is to compute...

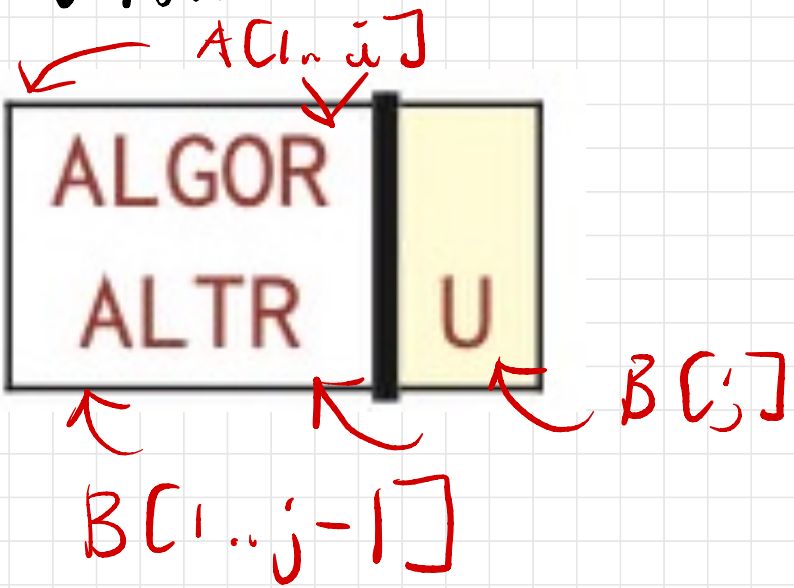
Edit distance of $A[1..m]$ + $B[1..n]$ is...

Edit(m, n)

Specification

For positive i & j, the "last edit" can be...

Insertion:



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

Deletion:

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

Substitution (?):

$$[A[i] \neq B[j]] = \begin{cases} 1 & \text{if } A[i] \neq B[j] \\ 0 & \text{o.w.} \end{cases}$$

$$Edit(i,j) = [A[i] \neq B[j]] + Edit(i-1, j-1)$$

$$Edit(0,j) = j$$

$$Edit(i,0) = i$$

$$(\Rightarrow Edit(0,0) = 0)$$

$$Edit(i,j) =$$

$$\begin{cases} j & \text{if } i = 0 \\ i & \text{if } j = 0 \\ \min \begin{cases} 1 + Edit(i, j-1) \\ 1 + Edit(i-1, j) \\ [A[i] \neq B[j]] + Edit(i-1, j-1) \end{cases} & \text{(o.w.)} \end{cases}$$

Subproblems:

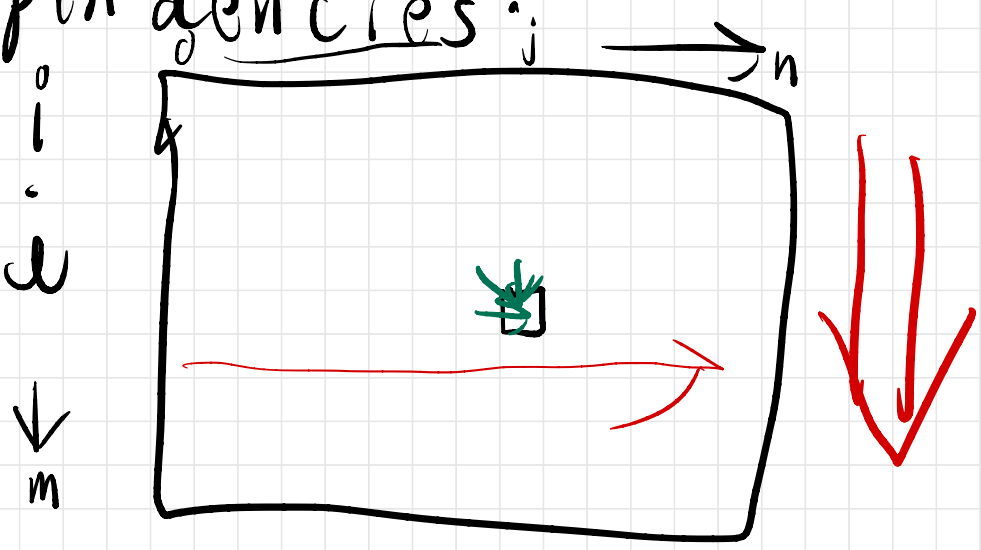
$0 \leq i \leq m$

$0 \leq j \leq n$

Data structure:

$Edit[0..m, 0..n]$

Dependencies:



Evaluation order:

"row major order"

Space: $O(mn)$

Time: $O(1) \cdot O(mn)$

$= O(mn)$

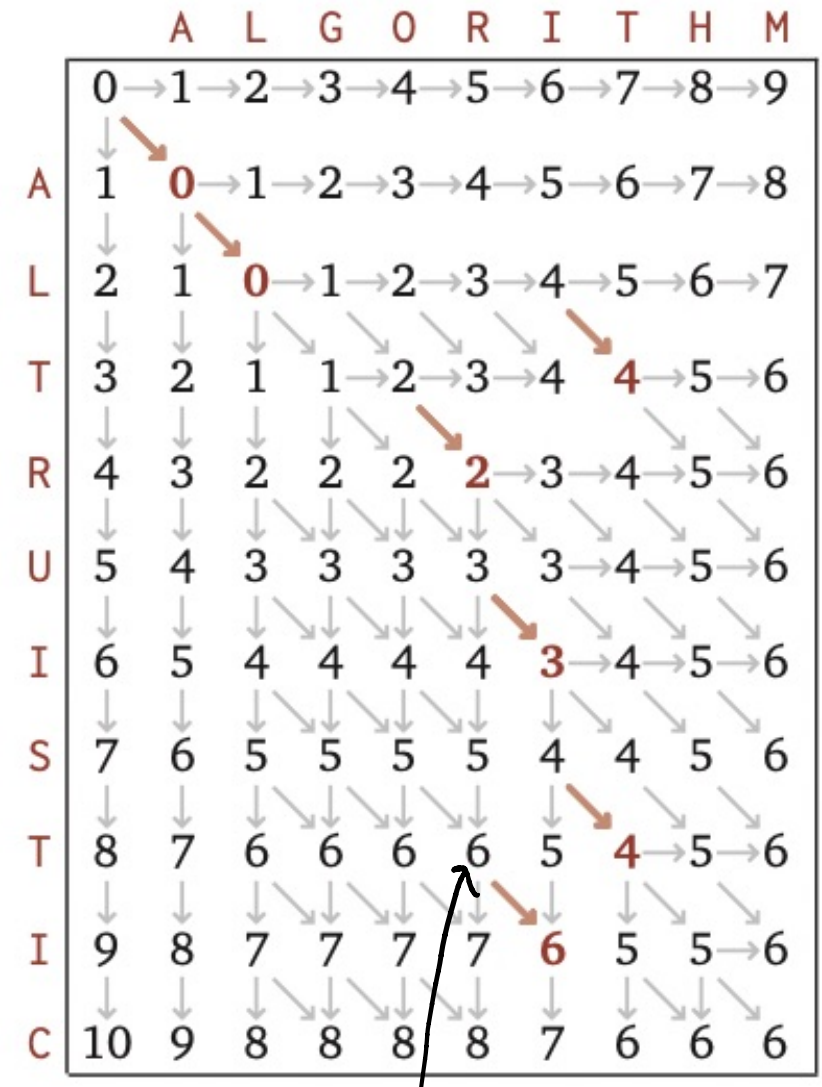
```
EDITDISTANCE(A[1..m], B[1..n]):
  for j ← 0 to n
      Edit[0, j] ← j

  for i ← 1 to m
      Edit[i, 0] ← i
      for j ← 1 to n
          ins ← Edit[i, j − 1] + 1
          del ← Edit[i − 1, j] + 1
          if A[i] = B[j]
              rep ← Edit[i − 1, j − 1]
          else
              rep ← Edit[i − 1, j − 1] + 1
          Edit[i, j] ← min {ins, del, rep}

  return Edit[m, n]
```

[Wagner & Fischer '74]

(Midterm 1 in two weeks)



$Edit(i, j)$

$Edit(8, 5) = 6$

→ answer from insert

↓ from delete

↘ from sub.

Follow arrows backwards from (m, n) to get best edit sequence.

$O(m+n)$ addition time.