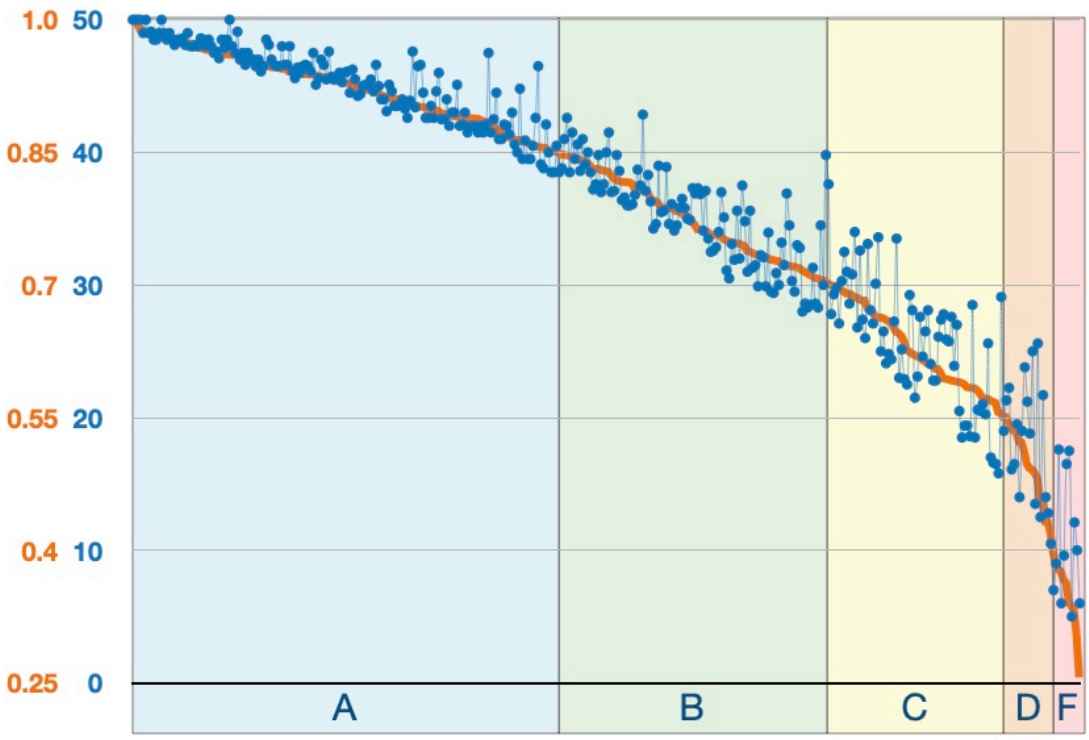
Orange line — computed course average
Blue dots — Midterm 1 scores

SOFAR



Dynamic Programming

- Aim for recursive backtracking
  brute force + recursion
  - Describe the problem you're trying to solve
    - What do I need to remember about past decisions? — INPUT
    - What am I trying to figure out about future decisions? — OUTPUT
  - Design a recursive algorithm
    - What is the next decision?
- Removing redundancy adding efficiency
  - Memoization structure
  - Evaluation order
  - Analyze time

EDIT DISTANCE

insertions
deletions
replacements

ALGORITHMS
ALTRUISTIC

libcurses

Del Ins Rep

ALGORITHMS
ALTRUISTIC

0 0 1 1 0 1 0 1 0 1 1 1

We're looking for a sequence of pairs...

ALGORI | THMS
ALTRUIS | T IC
(past) (future)

Need to remember:
chars in each string unprocessed

Let Edit(i,j) = edit distance between prefix A[1..i] and prefix B[1..j]

Recursive question: What's in the last column?
What's the last op?

Ins

ALGORS / ALTRUI | S

Edit(i,j-1)+1

Del

ALGOR / ALTRUIS | S

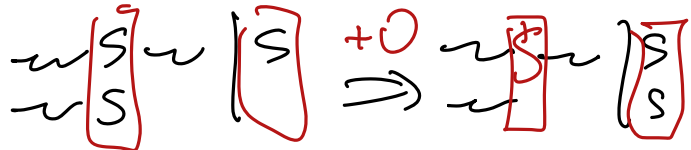
Edit(i-1,j)+1

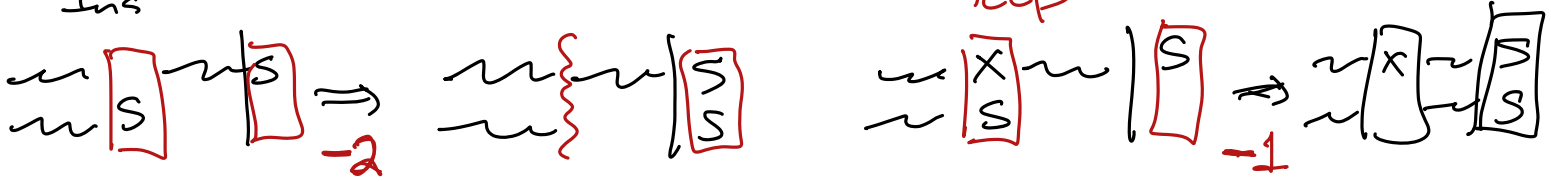
Rep

ALGOR / ALTRUI | S / S

Edit(i-1,j-1) +~~1~~ 0

min





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

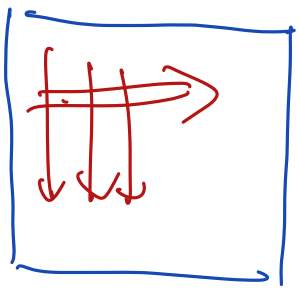
0..m 0..n

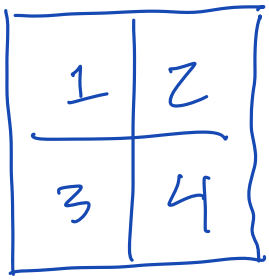
Iverson bracket
1 if $A[i] \neq B[j]$
0 otherwise

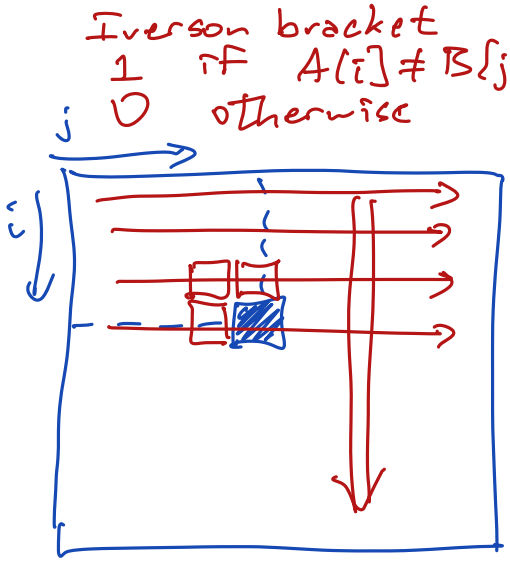
Memoization

Data Structure = 2d array
Edit[0..m, 0..n]

Eval Order







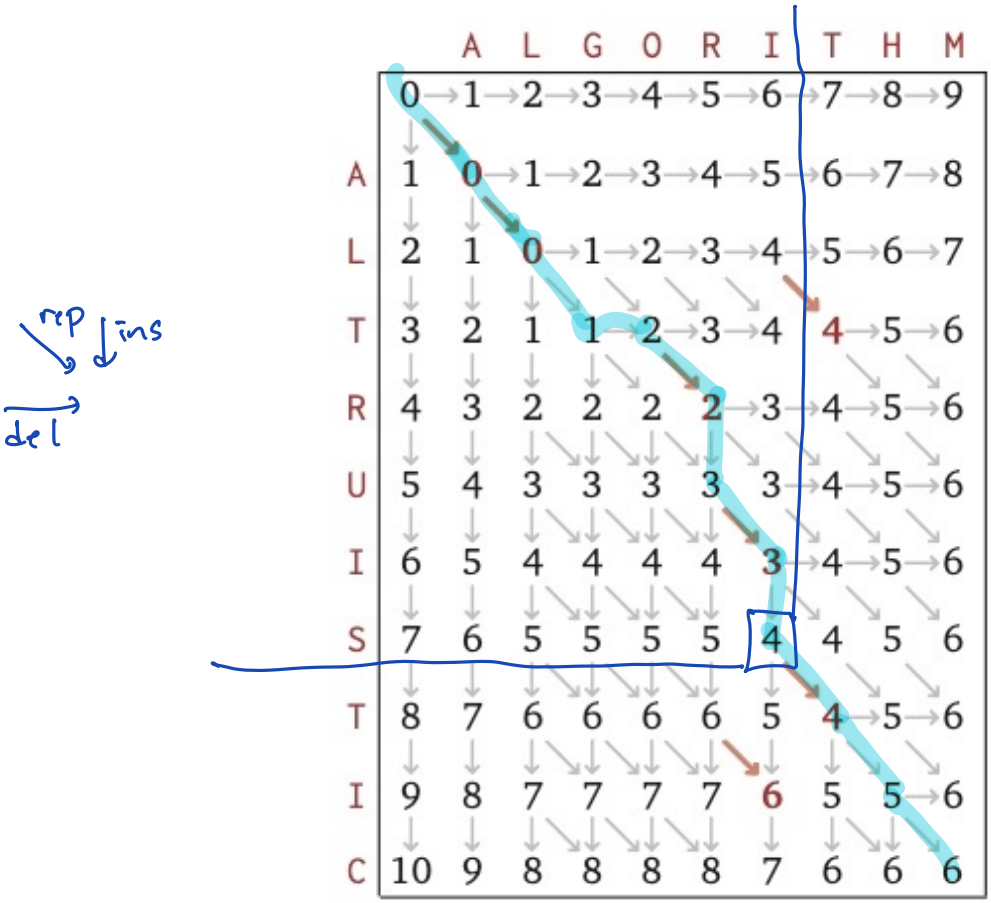

```
EditDistance(A[1..m], B[1..n]):
  for j ← 0 to n
      Edit[0, j] ← j

  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]
```

O(mn) time

recurrence


rep
ins
del
A L G O R I T H M
0 1 2 3 4 5 6 7 8 9
A 1 0 1 2 3 4 5 6 7 8
L 2 1 0 1 2 3 4 5 6 7
T 3 2 1 1 2 3 4 4 5 6
R 4 3 2 2 2 2 3 4 5 6
U 5 4 3 3 3 3 3 4 5 6
I 6 5 4 4 4 4 3 4 5 6
S 7 6 5 5 5 5 4 4 5 6
T 8 7 6 6 6 6 5 4 5 6
I 9 8 7 7 7 7 6 5 5 6
C 10 9 8 8 8 8 7 6 6 6