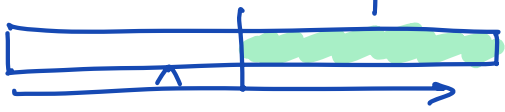
Dynamic Programming

① Aim for recursive backtracking algorithm

brute force + recursion

— What kind of subproblems?



Seq: suffix
LIS: suffix + one index

- What do I need to remember about past decisions?

- What am I trying to figure out about future decisions?

30% → English description of recursive problem

40% → Develop a recurrence / backtracking

Ex: LIS(i,j) is length of longest increasing subseq of A[j..n] all bigger than A[i]

$$LIS(i,j) = \begin{cases} 0 & \text{if } j > n \\ \max\{LIS(i,j+1),\ 1 + LIS(j,j+1)\} & \text{if } A[i] < A[j] \\ LIS(i,j+1) & \text{if } A[i] \geq A[j] \end{cases}$$

---

② Removing redundancy → make it fast

— choose memo data structure

typically array

30%

— Evaluation order

— time

A = ALGORITHMS

B = ALTRUISTIC

Edit Distance

minimum # of
insertions
deletions
replacements
to edit A into B

0 0 1 1 0 1 0 1 0 1 1 1 → 7

ALGORITHMS
ALTRUISTIC

insertion
deletion
replacement

what's in the last column?

Subproblems:

ALGORI | THMS
ALTRUIS | TIC

future ← past

Edit(i,j) = edit distance from A[1..i] to B[1..j]

Final answer Edit(m,n) ≤ m+n

ALGORI | THMS
ALTRUIS | TIC

Edit(i-1,j-1) + [A[i] ≠ B[j]]

!!(expr)

ALGORI | THMS
ALTRUIS | TIC

Edit(i-1,j) + 1

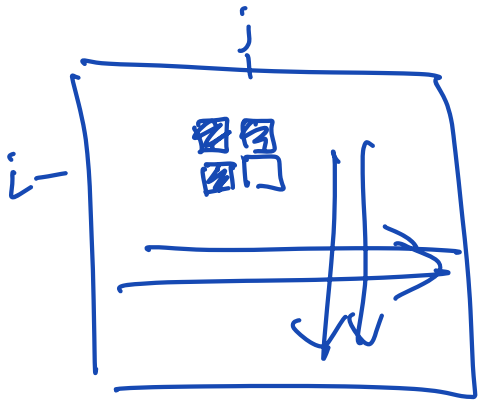
ALGORI | THMS
ALTRUIS | TIC

Edit(i,j-1) + 1

min

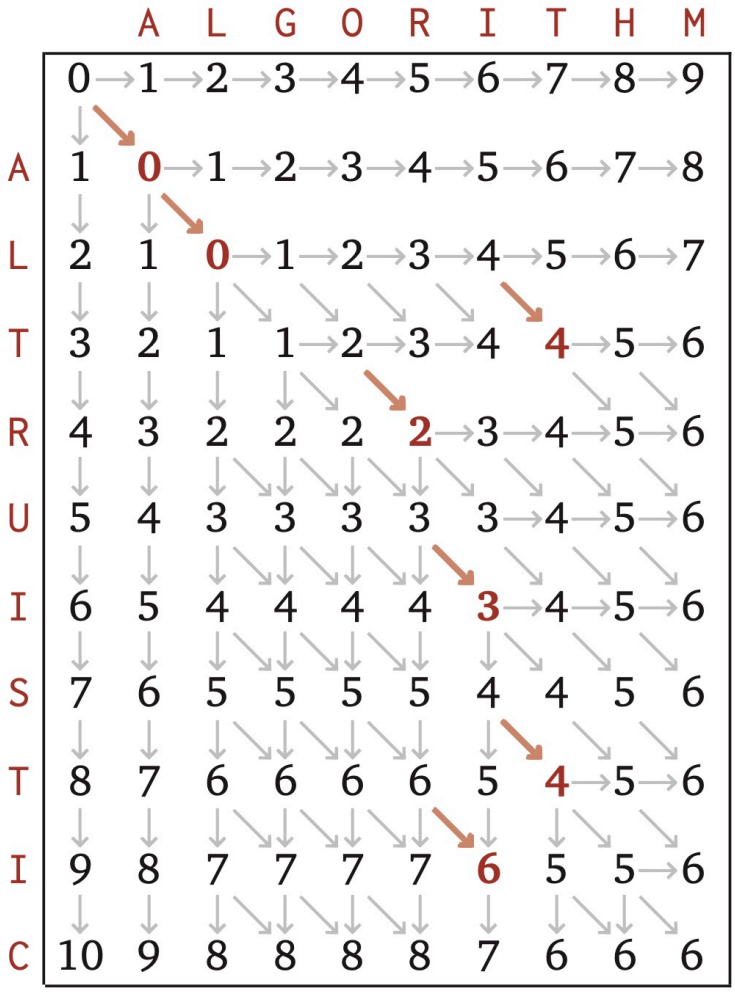
$$Edit(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \begin{cases} Edit(i,j-1)+1 \\ Edit(i-1,j)+1 \\ Edit(i-1,j-1)+[A[i] \neq B[j]] \end{cases} & \text{otherwise} \end{cases}$$

ins
del
rep

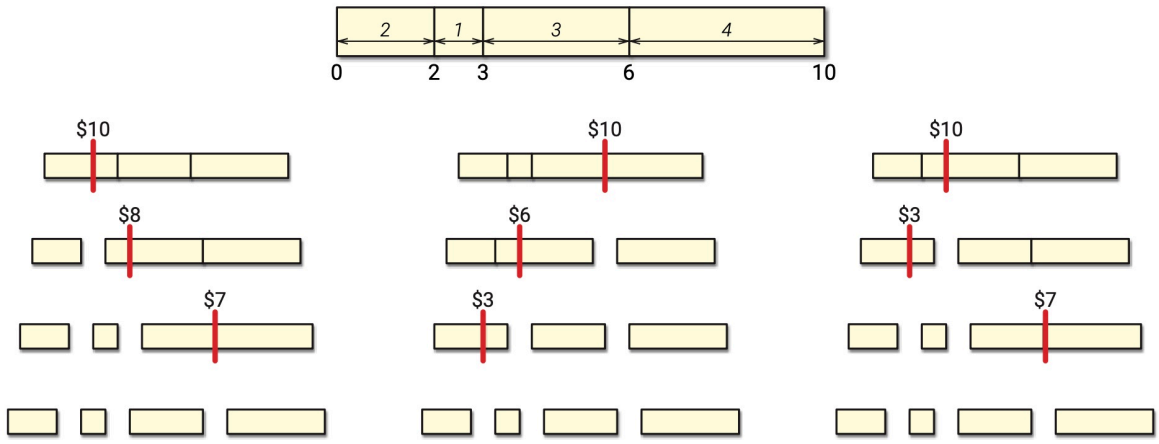


$O(mn)$ time

```
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]
```




```
A L G O R I   T H M
A L T R U I S T I C

A L G O R   I   T H M
A L   T R U I S T I C

A L G O R   I   T H M
A L T   R U I S T I C
```



$$OptCost(i,k) = \begin{cases} 0 & \text{if } i = k \\ \sum_{j=i}^{k} L[j] + \min_{i \le r < k} \left\{ \begin{matrix} OptCost(i, r-1) \\ + OptCost(r, k) \end{matrix} \right\} & \text{otherwise} \end{cases}$$