Today

- HW 2 is done but not entered
- HW3 link didn't go live
- Perusall due Wed.

Edit Distance

The minimum number of deletions, insertions, or substitutions of letters to transform between two strings.

Ex: F O O D

+1 +1 +1 insert +1

M O N E Y

Uses? - Spell checker
- bioinformatics

How to solve:

Aligning/matching will help:

A: A L G O R I T H M

+0 +0

B: A L T R U I S T I C

+1 +1 +1 +1 +1 +1

distance?

6

Example:

TGCATAT
$\downarrow$ delete last T
TGCATA
$\downarrow$ delete last A
TGCAT
$\downarrow$ insert A at the front
ATGCAT
$\downarrow$ substitute C for G in the third position
ATCCAT
$\downarrow$ insert a G before the last A
ATCCGAT

TGCATAT
$\downarrow$ insert A at the front
ATGCATAT
$\downarrow$ delete T in the sixth position
ATGCAAT
$\downarrow$ substitute G for A in the fifth position
ATGCGAT
$\downarrow$ substitute C for G in the third position
ATCCGAT

Alignment matrix:

A	T	-	G	T	T	A	T	-
A	T	C	G	T	-	A	-	C

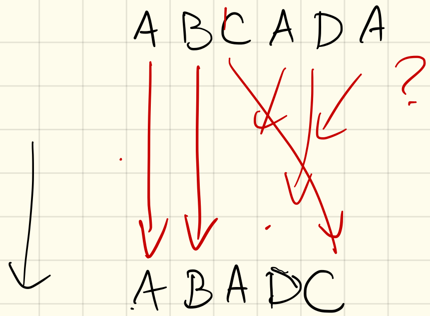
(at most m+n columns)

Another way:
Write # of repititions:

		0	1	2	2	3	4	5	6	7	7
v	=		A	T	-	G	T	T	A	T	-
			|	|		|	|		|		
w	=		A	T	C	G	T	-	A	-	C
		0	1	2	3	4	5	5	6	6	7

Don't be greedy!

The temptation is to do this as you go:



edit distance?

Idea: try matching, or not

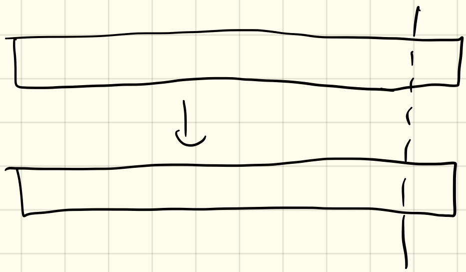
try both, pay costs that depend on letters

Recursive formulation:

If I align like this, can observe:

If you delete last (aligned) column, the rest will still be optimal for shorter substrings edit distance.

Why?



Turning this into a matrix!

Let EDIT(A[1..m], B[1..n])
be edit distance b/t A & B.

When we choose how to align, 3 possibilities:

- insertion:

- deletion:

- substitution:

$$\mathit{Edit}(A[1..m], B[1..n]) = \min \left\{ \begin{array}{l} \mathit{Edit}(A[1..m-1], B[1..n]) + 1 \\ \mathit{Edit}(A[1..m], B[1..n-1]) + 1 \\ \mathit{Edit}(A[1..m-1], B[1..n-1]) + \big[A[m] \neq B[n]\big] \end{array} \right\}$$

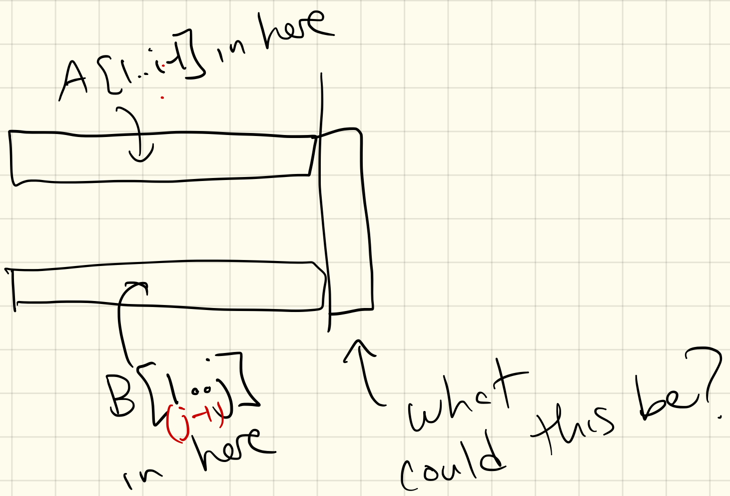
Turning this into a proper recursion:

Let $\text{EDIT}(i,j) :=$ edit distance between:

$A[1..i]$

$B[1..j]$

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



Give me 2 strings:

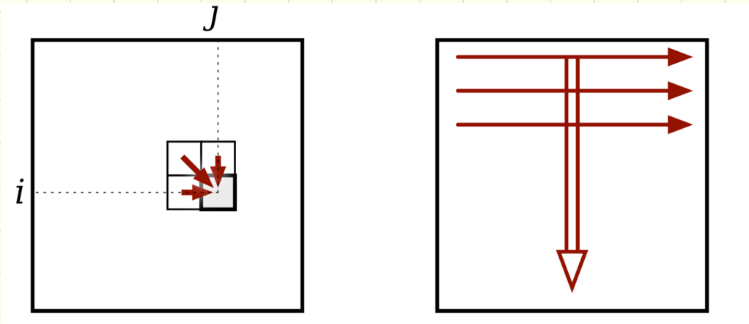
$$
\boxed{Edit(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \left\{ \begin{array}{l} Edit(i-1,j)+1, \\ Edit(i,j-1)+1, \\ Edit(i-1,j-1)+\left[A[i] \neq B[j]\right] \end{array} \right\} & \text{otherwise} \end{cases}}
$$

Now, don't bother analyzing the recursion.

(It's awful!)

Instead, be smart: memoize!

Table:



Algorithm:

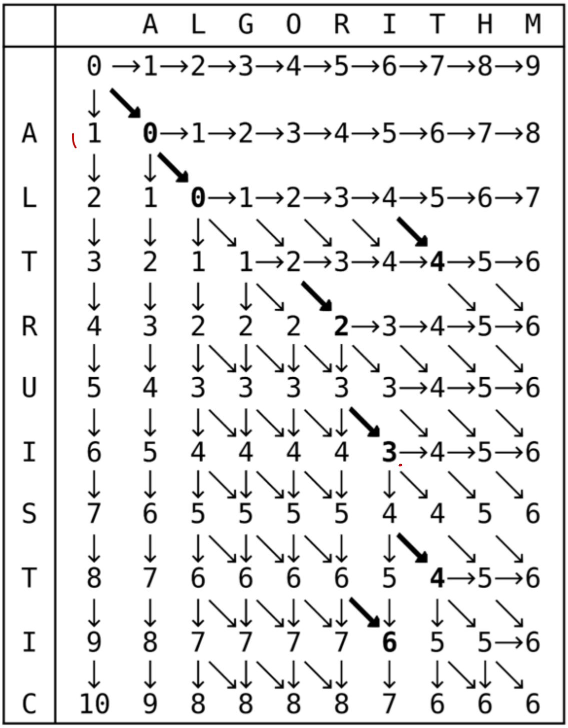
```
EDITDISTANCE(A[1..m], B[1..n]):
    for j ← 1 to n
        Edit[0, j] ← j

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

    return Edit[m, n]
```

Runtime:

Example:

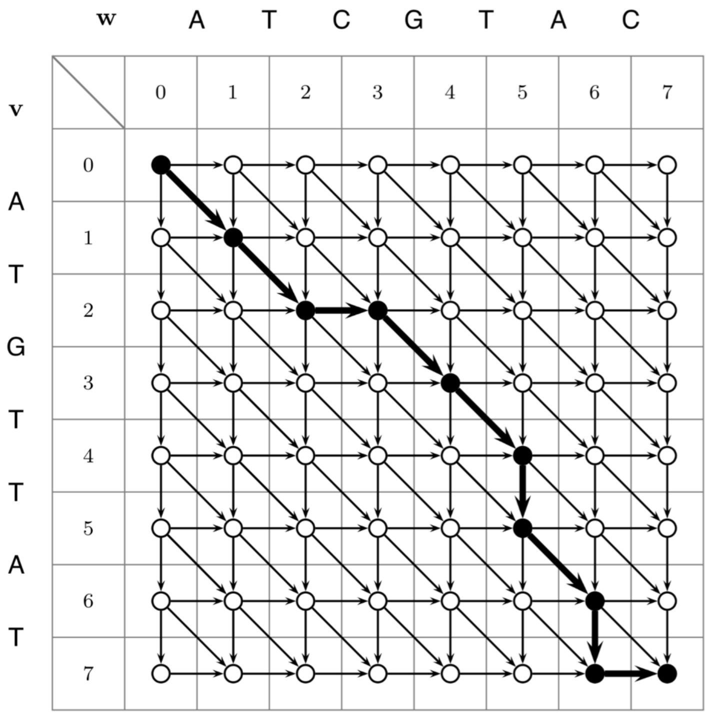


The memoization table for *Edit*(ALGORITHM, ALTRUISTIC)

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

Another: (DNA Example)

$$
\begin{array}{ccccccccccc}
 & & 0 & 1 & 2 & 2 & 3 & 4 & 5 & 6 & 7 & 7 \\
\mathbf{v} & = & & \mathsf{A} & \mathsf{T} & - & \mathsf{G} & \mathsf{T} & \mathsf{T} & \mathsf{A} & \mathsf{T} & - \\
 & & & | & | & & | & | & & | & & \\
\mathbf{w} & = & & \mathsf{A} & \mathsf{T} & \mathsf{C} & \mathsf{G} & \mathsf{T} & - & \mathsf{A} & - & \mathsf{C} \\
 & & 0 & 1 & 2 & 3 & 4 & 5 & 5 & 6 & 6 & 7
\end{array}
$$



$$
\begin{array}{ccccccccc}
\searrow & \searrow & \rightarrow & \searrow & \searrow & \downarrow & \searrow & \downarrow & \rightarrow \\
A & T & - & G & T & T & A & T & - \\
A & T & C & G & T & - & A & - & C
\end{array}
$$

Next: Subset Sum

Given a set X of positive integers and a target value t, is there a subset of X which sums to t?

Recall our (exponential) backtracking.

Formalize this: recursion!

$$T(\underbrace{X}_{X[1..n]}, t) = \begin{cases} \text{include } X[1]: \\ T(X[2..n], t - X[1]) \\ \text{not}: \\ T(X[2..n], t) \end{cases}$$

<u>SubsetSum($X[1..n], T$):</u>
  if $T = 0$
    return True
  else if $T < 0$ or $n = 0$
    return False
  else
    return $\big($SubsetSum($X[1..n-1], T$) $\vee$ SubsetSum($X[1..n-1], T - X[n]$)$\big)$

Can we do DP?

In this chapter:

$$SS(i,t) = \begin{cases} \text{TRUE} & \text{if } t = 0 \\ \text{FALSE} & \text{if } t < 0 \text{ or } i > n \\ SS(i+1,t) \vee SS(i+1,t-X[i]) & \text{otherwise} \end{cases}$$

Or:

$$SS(i,t) = \begin{cases} \text{TRUE} & \text{if } t = 0 \\ \text{FALSE} & \text{if } i > n \\ SS(i+1,t) & \text{if } t < X[i] \\ SS(i+1,t) \vee SS(i+1,t-X[i]) & \text{otherwise} \end{cases}$$

How to memoize?