Dynamic Programming II

Sept 9, 2025

Erickson's Advice : Always compose a human-language (English) specification of the underlying recursive subproblems.

Let's do that for the Edit Distance problem.

Problem: Given two strings over a finite alphabet (say A B C . . . Z), find the fewest number of operations required to transform one string into the other.

operations : letter-insert, -delete, -substitute.

FOOD → MOOD → MOND → MONED → MONEY.

4 steps.

Observation about a transformation of the problem :

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

gap representation

The solution is essentially found once we have lined the two strings up in columns that minimize the number of mismatched columns.

Defining the recursive subproblems (in English).

For any two input strings $A[1..n]$ and $B[1..m]$:

Let $Edit(i, j)$ be edit distance between

$A[1..i]$ and $B[1..j]$.

The solution we seek is $Edit(n, m)$

That is not yet a solution, and we don't even yet know if it is a fruitful rec-subproblem defn — we'll only know that if we can do this next bit.

Showing how to construct the overall solution from recursive-subproblems:

ALGOR ALTRU

$\exists$ 3 possibilities for the final column:

1.	2.	3.
ALGOR ALTR U	ALGO R ALTRU	ALGO R ALTR U
insertion	deletion	substitution.

Using our recursive subproblem formulation:

$\text{Edit}(i, j) =$ $\text{Edit}(i, j-1)$ $+ 1$ insertion	$\text{Edit}(i, j) =$ $\text{Edit}(i-1, j)$ $+ 1$ deletion	$\text{Edit}(i, j) =$ $\text{Edit}(i-1, j-1)$ $+1$ substitution or $+0$ if match

Now we simply minimize over these 3 options:

$$\text{Edit}(i, j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \begin{cases} \text{Edit}(i, j-1) + 1 \\ \text{Edit}(i-1, j) + 1 \\ \text{Edit}(i-1, j-1) + \underbrace{[A[i] \neq B[j]]}_{\substack{1 \text{ if no match} \\ 0 \text{ if match}}} \end{cases} & \text{otherwise} \end{cases}$$

Now we have the recurrence that yields a dynamic programming solution:

Subproblems: $\text{Edit}(i, j)$ is the subproblem of the (min) edit distance between prefixes of $A[1..n]$ and $B[1..m]$; ie $A[1..i]$, $B[1..j]$.

$0 \leq i \leq n \qquad 0 \leq j \leq m$

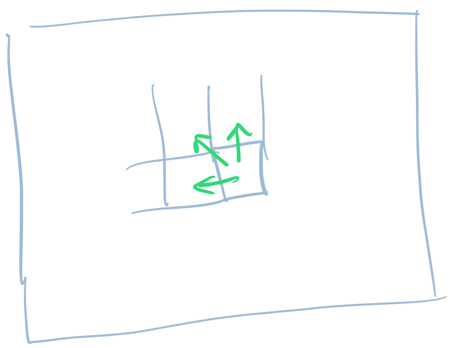
Memoization structure

Edit[n][m] – allows us to store Edit(i,j) in Edit[i,j]

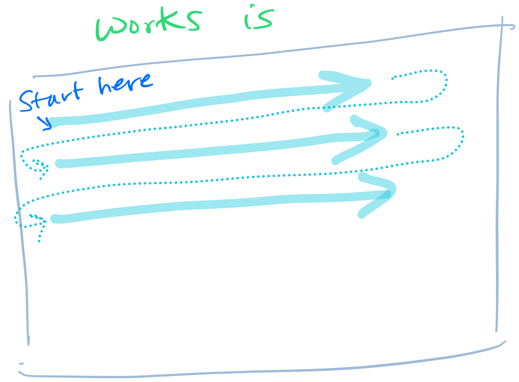
dependencies

Edit[i,j] depends on Edit[i,j-1], Edit[i-1,j], Edit[i-1,j-1].

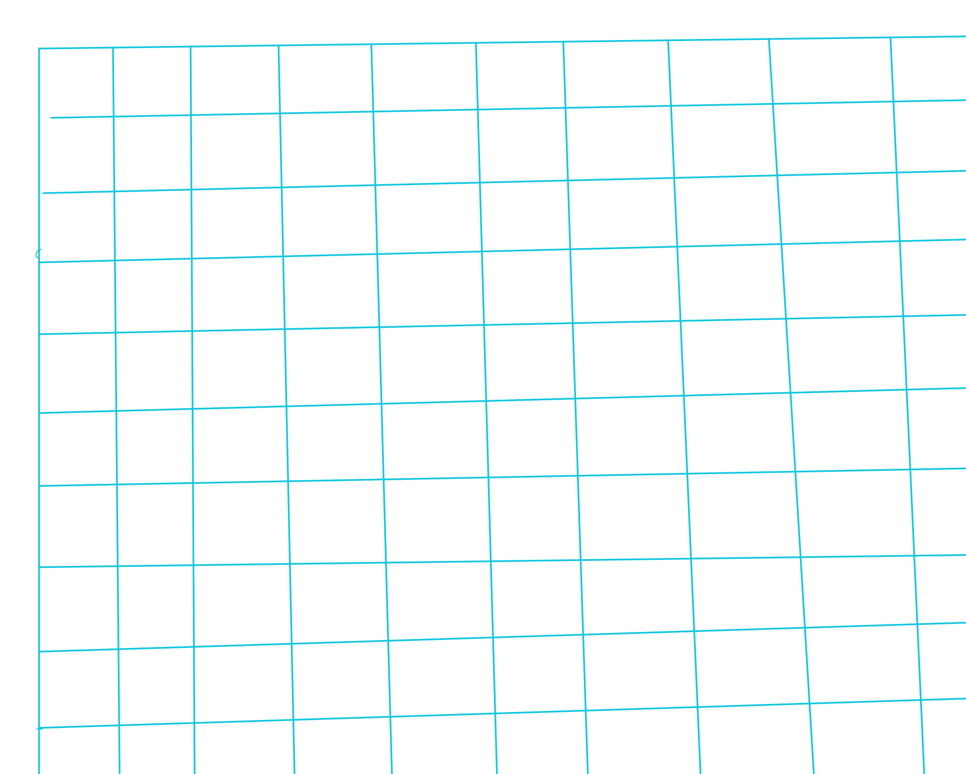
Edit



∴ an evaluation order that works is



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



to

from

	ε	A	L	G	O	R	I	T	H	M
ε	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