Edit Distance between two strings is min # insertions, deletions, & substitutions of single characters to turn one string into the other.

FOOD → MOOD → MOND
→ MONED
→ MONEY

E.D. ≤ 4

Levenshtein (coding theory),
Ulam (biological sequences)
Vintsyuk (speech recognition)

Input: two string
$A[1..m]$, $B[1..n]$

Output: Their edit distance.

```
F O O   D
M O N E Y
```

Imagine edits by stacking strings. Each column is an edit.

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

What should we do for the last column?

Use backtracking to try all options.

The rest of the columns should represent the edit distance of what remains!

So recursive subproblems are prefixes...
encoded as their final indices
(i.e. their lengths)

Edit(i,j): edit distance for $A[1..i]$ & $B[1..j]$.
We want to compute Edit(m,n).

Recurrence:

Suppose $i > 0$, $j > 0$. If our optimal last column is an...

Insertion:

ALGOR	
ALTR	U

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

Deletion:

ALGO	R
ALTRU	

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

Substitution(?):

ALGO	R
ALTR	U

ALGO	R
ALT	R

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

$$[A[i] \neq B[j]]$$

indicator variable
$=1$ if proposition
is true &
0 o.w.

$\text{Edit}(i,j)$ uses min of those three expressions.

$$\text{Edit}(0,j) = j$$
$$\text{Edit}(i,0) = i$$
$$\Rightarrow \text{Edit}(0,0) = 0$$

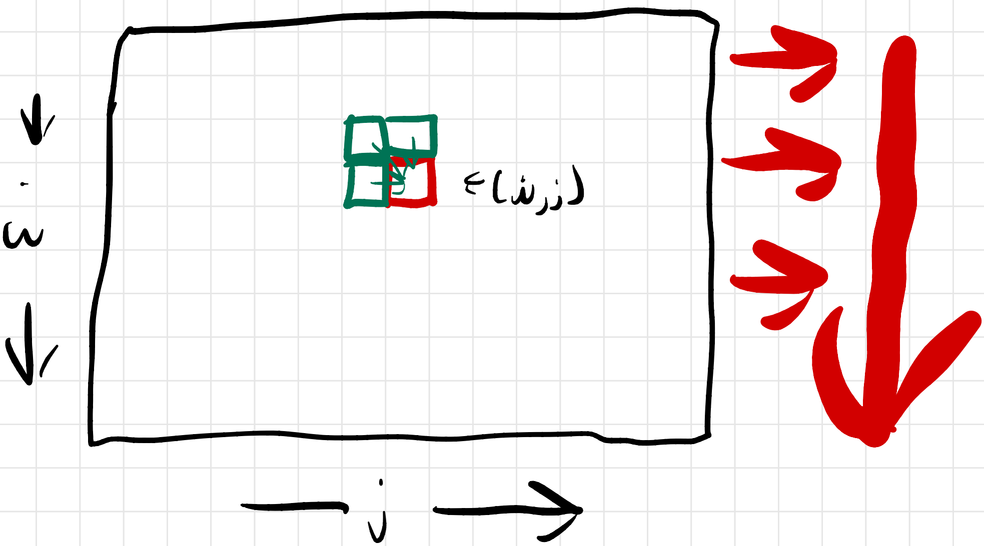
$$
Edit(i,j) = \begin{cases} j & \text{if } i = 0 \\ i & \text{if } j = 0 \\ \min \begin{cases} 1 + Edit(i, j-1), \\ 1 + Edit(i-1, j), \\ [A[i] \neq B[j]] + Edit(i-1, j-1) \end{cases} & \text{o.w.} \end{cases}
$$

Dynamic Programming

Subproblems: $0 \le i \le m$
$0 \le j \le n$

Data structure: 2D Array
Edit[0..m, 0..n]

Dependencies:



Eval 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.

		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

Can trace reasons for each optimal choice to see sequence of edits in $O(m+n)$ additional time.

Optimal Binary Search Trees

Input: Array of <u>keys</u> $A[1..n]$, that are sorted

<u>frequencies</u> $f[1..n]$

Key $A[i]$ will be sought after $f[i]$ times.

How to minimize <u>total</u> search time in a BST over A.

(<u>A</u> is sorted. Not f)

For a given BST T:

$v_1, v_2, \ldots, v_n$ in order

v_i store $A[i]$

$Cost(T, f[1..n]) ::=$

$$\sum_{i=1}^{n} f[i] \cdot \#\text{ancestors of } v_i \text{ in } T$$

(root has 1 ancestor, itself)

T has a root, a left subtree, right subtree...

r : root key

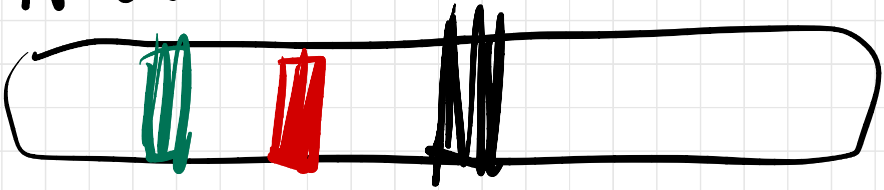
$$\mathrm{Cost}(T, f) = \sum_{i=1}^{n} f[i] + \sum_{i=1}^{r-1} f[i] \cdot \#\text{anc in left}(T) + \sum_{i=r+1}^{n} f[i] \cdot \#\text{anc. in right}(T)$$

$$= \sum_{i=1}^{n} f[i] + \mathrm{Cost}(\text{left}(T), f[1..r-1]) + \mathrm{Cost}(\text{right}(T), f[r+1..n])$$

$$\mathrm{Cost}(T, f[1..0]) = 0$$

Idea: Guess root key & recuse on smaller & greater keys.

Recursive subsets are contiguous!



OptCost(i,k): optimal cost of any tree over $A[i..k]$.

$$OptCost(i,k) = \begin{cases} 0 & \text{if } i > k \\ \underbrace{\left(\sum_{j=i}^{k} f[j]\right)}_{\text{cost of touch root over all searches}} + \min_{\underbrace{i \le r \le k}_{\text{guess for root key}}} \{OptCost(i, r-1) + OptCost(r+1, k)\} & \text{o.w.} \end{cases}$$