Last Time

- Midterm Exam
  - Mean: 90.5
  - Std. Dev.: 9.63
  - Median: 94
Dynamic Programming
Identify the three numbers that have the greatest product from the following set.

[5, 1, 0.5, 10, 2]
Identify the three numbers that have the greatest product from the following set.

\[5, 1, 0.5, 10, 2\]
\[-5, -1, -0.5, 10, 2\]
Identify the three numbers that have the greatest product from the following set.

\[ [5, 1, 0.5, 10, 2] \]
\[ [-5, -1, -0.5, 10, 2] \]

- Find the “best” solution to some problem.
Identify the three numbers that have the greatest product from the following set.

[5, 1, 0.5, 10, 2]
[-5, -1, -0.5, 10, 2]

- Find the “best” solution to some problem.
- There are many valid solutions – here any set of 3 numbers is a solution
- Find the solution which has a maximum or minimum value – Efficiently.
Identify the three numbers that have the greatest product from the following set.

\[5, 1, 0.5, 10, 2\]
\[-5, -1, -0.5, 10, 2\]

- Find the “best” solution to some problem.
- There are many valid solutions – here any set of 3 numbers is a solution.
- Find the solution which has a maximum or minimum value – Efficiently.
- Search over the space of (partial) solutions.
Dynamic Programming

- The solution of the problem is made up of solutions to sub-problems.
- Similar to Divide and Conquer.
- Subproblems are frequently reused.
- **Save the solutions to subproblems**
Recall **Fibonacci Sequence**

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . .

Recursively.

$Fib(1) = 1$

$Fib(2) = 1$

$Fib(n) = Fib(n - 1) + Fib(n - 2)$
Recursive Fib($i$)

Fib($i$)

if $i = 1$ or $i = 2$ then
    return 1
end if

return Fib($n-1$) + Fib($n-2$)
How many times is $Fib(n)$ called?
Fib(3)

Fib(3)

Fib(2)  Fib(1)

1  1

1  2
Fib(4)

1
2
1
2
1
1
Fib(4)
Fib(3)
Fib(2)
Fib(4)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
1
2
1
2
1
2
1
Fib(n)

1 = 2^0
2 = 2^1
4 = 2^2
8 = 2^3
\( \text{Fib}(n) = O(2^n) \)

But we recalculate \( \text{Fib}(n-2) \) when calculating \( \text{Fib}(n) \) and \( \text{Fib}(n-1) \)
Fibonacci Dynamic Programming

Fib-DP(n)

\[
\begin{align*}
c[1] &= 1 \\
c[2] &= 1 \\
\text{for } i &\leftarrow 1..n \text{ do} \\
&\quad c[i] \leftarrow c[i - 1] + c[i - 2] \\
\text{end for} \\
\text{return } c[n]
\end{align*}
\]

- \(O(n)\)


- **Input:** \( n \) denominations of coins \( 1 = v_1 < v_2 < \ldots < v_n \)
- **Problem:** Make change for amount of money \( C \) using as few coins as possible.
- **Note:** all of \( v_i \) and \( C \) are positive integers.
Making Change – Dynamic Programming

\( M(j) \) = the minimum number of coins required to make change for an amount of money \( j \).

\[
M(j) = \min_i \{ M(j - v_i) \} + 1
\]

The smallest number of coins required to make \( j \), is the smallest number required to make \( j - v_i \), plus one.

(Aside: How would you prove it?)

To calculate, start from the smallest amount, 1, and build up a table \( M \).
Making Change Example

\[ v_1 = 1 \]
\[ v_2 = 3 \]
\[ v_3 = 4 \]
\[ C = 10 \]
Making Change Example

\[ v_1 = 1 \]
\[ v_2 = 3 \]
\[ v_3 = 4 \]
\[ C = 10 \]

\[ M[i] \quad [1] \]
Making Change Example

\[ v_1 = 1 \]
\[ v_2 = 3 \]
\[ v_3 = 4 \]
\[ C = 10 \]

M[i] \[\begin{array}{c}
  1 \\
  2 
\end{array}\]
$v_1 = 1$
$v_2 = 3$
$v_3 = 4$
$C = 10$
$M[3] = 1$

M[i] 1 2 1
Making Change Example

\[ v_1 = 1 \]
\[ v_2 = 3 \]
\[ v_3 = 4 \]
\[ C = 10 \]
\[ M[4] = 1 \]

\[ M[i] \quad 1 \quad 2 \quad 1 \quad 1 \quad 1 \]
Making Change Example

\[ \nu_1 = 1 \]
\[ \nu_2 = 3 \]
\[ \nu_3 = 4 \]
\[ C = 10 \]

\[
\begin{array}{cccc}
M[i] & 1 & 2 & 1 & 1 & 2
\end{array}
\]
\( \nu_1 = 1 \)
\( \nu_2 = 3 \)
\( \nu_3 = 4 \)
\( C = 10 \)
\( v_1 = 1 \)
\( v_2 = 3 \)
\( v_3 = 4 \)
\( C = 10 \)
\[ v_1 = 1 \]
\[ v_2 = 3 \]
\[ v_3 = 4 \]
\[ C = 10 \]
Making Change Example

\[ \nu_1 = 1 \]
\[ \nu_2 = 3 \]
\[ \nu_3 = 4 \]
\[ C = 10 \]
Making Change Example

\[ v_1 = 1 \]
\[ v_2 = 3 \]
\[ v_3 = 4 \]
\[ C = 10 \]
Minimum-Change(C, v)

\[ M[0] \leftarrow 0 \{ M[j] = \infty \text{ where } i < 0 \} \]

for \( j \leftarrow 1..C \) do

\[ min \leftarrow \infty \]

for \( i \leftarrow 1..|v| \) do

if \( M[j - v[i]] + 1 < min \) then

\[ min \leftarrow M[j - v[i]] + 1 \]

end if

end for

return \( M[C] \)

end for
Elements of Dynamic Programming

- Optimal Substructure
- Overlapping Subproblems
- Reconstruction of an Optimal Solution
Optimal Substructure

A problem demonstrates **optimal substructure** if an optimal solution is comprised of optimal solutions to subproblems.

**Identifying “optimal substructures”**

- Show that the solution involves making a **choice**, leaving a subproblem or set of subproblems to be solved.

- Suppose that you are given the choice that leads to an optimal solution. (For now don’t worry how to make that choice.)

- Determine which subproblems remain after the optimal choice, and how to characterize the space of subproblems.

- Show that the solutions to subproblems used in the optimal solution must also be optimal.
Identifying “optimal substructures”

- Show that the solutions to subproblems used in the optimal solution must also be optimal.
  - Suppose each subproblem’s solution is not optimal and derive a contradiction.
  - Specifically, that if the subproblem solution isn’t optimal, then the overall solution isn’t optimal.
- A “cut-and-paste” technique: Argue that you can “cut” the suboptimal solution, and “paste” an optimal subproblem solution leading to a better overall solution. Since we assume the overall solution is optimal, there is a contradiction.
Overlapping Subproblems

Dynamic Programming leads to efficient solutions to problems when subproblems are frequently reused. Dynamic Programming is similar to divide-and-conquer.

- Identify subproblems.
- Solve subproblems.
- Combine solutions.

For divide-and-conquer the subproblems are unique.

In Dynamic Programming the subproblems are frequently repeated. The efficiency is introduced by reusing solutions. In practice, the solutions to subproblems are stored in a table.
Reconstruction of Optimal Solution

The last component of a Dynamic Programming algorithm is the reconstruction of a solution.

The dynamic programming table contains the value that is being optimized. Reconstructing the set of choices made to arise at the optimal solution is not always trivial.

This typically involves storing the choices made in addition to the value either in the DP table, or in a second table of the same dimensions.
How different are the strings **LEAD** and **LAST**?
How different are the strings **LEAD** and **LAST**?

Approach: Count the number of editing operations required to transform one to the other.
How different are the strings LEAD and LAST?

Approach: Count the number of editing operations required to transform one to the other.

Define three (or two) edit operations: Insert, Delete and Substitution*.

This is known as Minimum Edit Distance (MED) or Levenshtein Distance.
Minimum Edit Distance

How different are the strings **LEAD** and **LAST**?

Three.

One Deletion: LEAD $\rightarrow$ LAD

One Substitution: LAD $\rightarrow$ LAT

One Insertion: LAT $\rightarrow$ LAST
Dynamic Programming calculation of MED.

Optimal Substructure.

Two strings: \( s_1 \) and \( s_2 \).

\( s_1[1..i] \) and \( s_2[1..j] \) have a minimum edit distance of \( c \).

\[ d(s_1[1..i], s_2[1..j]) = c \]

Claim: \( c = \) the smallest of 4 values.

1. \( d(s_1[1..i - 1], s_2[1..j - 1]) \) if \( s_1[i] == s_2[j] \) - equivalence
2. \( d(s_1[1..i - 1], s_2[1..j - 1]) + 1 \) if \( s_1[i]! = s_2[j] \) - substitution
3. \( d(s_1[1..i], s_2[1..j - 1]) + 1 \) - insertion
4. \( d(s_1[1..i - 1], s_2[1..j]) + 1 \) - deletion

Homework: prove this claim.
Recursive MED

This minimization leads to a potential solution.

\[ d(s_1, s_2) \]

\[
\text{if } s_1.\text{size} = 0 \text{ and } s_2.\text{size} = 0 \text{ then}
\]
\[
\quad \text{return } 0
\]
\[
\text{end if}
\]
\[
\text{if } s_1.\text{size} = 0 \text{ then}
\]
\[
\quad \text{return } d(s_1, s_2[1..s_2.\text{size} - 1]) + 1 \text{ delete}
\]
\[
\text{end if}
\]
\[
\text{if } s_2.\text{size} = 0 \text{ then}
\]
\[
\quad \text{return } d(s_1[1..s_1.\text{size} - 1], s_2) + 1 \text{ insert}
\]
\[
\text{end if}
\]
\[
\text{return } \text{Min}(d(s_1, s_2[1..s_2.\text{size} - 1]) + 1,
\]
\[
\quad d(s_1[1..s_1.\text{size} - 1], s_2) + 1,
\]
\[
\quad d(s_1[1..s_1.\text{size} - 1], s_2[1..s_2.\text{size} - 1]) + 1)
\]

Runtime?
Recursive MED

This minimization leads to a potential solution.

d(s₁, s₂)

if s₁.size = 0 and s₂.size = 0 then
  return 0
endif

if s₁.size = 0 then
  return d(s₁, s₂[1..s₂.size − 1]) + 1 delete
endif

if s₂.size = 0 then
  return d(s₁[1..s₁.size − 1], s₂) + 1 insert
endif

return Min(d(s₁, s₂[1..s₂.size − 1]) + 1,
           d(s₁[1..s₁.size − 1], s₂) + 1,
           d(s₁[1..s₁.size − 1], s₂[1..s₂.size − 1]) + 1)

Runtime? T(n) = 3(T(n-1)) + 1 = Θ(3ⁿ)
Rather than recurse starting at the full strings with a stopping condition, we can **build up** a solution to the final problem from smaller subproblems.

Construct a table.

Let $k = s_1.size$ and $l = s_2.size$

Goal: construct an $k + 1$-by-$l + 1$ table $M$, such that $M[i,j]$ is the minimum number of edits to convert $s_1[1..i−1]$ to $s_2[1..j−1]$.

If this goal is accomplished, then $d(s_1, s_2) = M[k + 1, l + 1]$
Example MED table

**Initialization:** Set top row and left column.

<table>
<thead>
<tr>
<th></th>
<th>∅</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Initialization:** Set top row and left column.

<table>
<thead>
<tr>
<th></th>
<th>∅</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i,j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>∅</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>∅</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Example MED table**

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

Propagation: Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1, M[i-1,j] + 1, M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

Propagation: Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
### Example MED Table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$. 

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Example MED table

**Propagation:** Loop over each cell in the table, comparing values $M[i-1,j-1] + 1$, $M[i-1,j] + 1$, $M[i, j-1] + 1$.

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$</th>
<th>L</th>
<th>A</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
So far we have said that Insertion, Deletion and Substitution all have the **same cost**.

However, we can set distinct costs for each operation.

In this case we compare the values:

- \( M[i-1,j-1] + \text{Substitution Cost} \)
- \( M[i, j-1] + \text{Insertion Cost} \)
- \( M[i-1, j] + \text{Deletion Cost} \)
So far we have said that Insertion, Deletion and Substitution all have the **same cost**.

However, we can set distinct costs for each operation.

In this case we compare the values:

- \( M[i-1,j-1] + \text{Substitution Cost} \)
- \( M[i, j-1] + \text{Insertion Cost} \)
- \( M[i-1, j] + \text{Deletion Cost} \)

This is what you will need to implement for Homework 7.

Write pseudocode for your algorithm.
Next time (10/29)
- Greedy Algorithms
For Next Class
- Chapter 16.1, 16.2