Applications

- "diff" utility – where do two files differ
- Version control & patch distribution – save/send only changes
- Molecular biology
  - Similar sequences often have similar origin and function
  - Similarity often recognizable despite millions or billions of years of evolutionary divergence

Recursive Solution

- Sub-problems: Edit distance problems for all prefixes of A and B that don’t include all of both A and B
- Let D(i,j) be the number of edits required to transform $a_1 \ a_2 \ ... \ \ a_i$ into $b_1 \ b_2 \ ... \ b_j$
- Clearly $D(0,0)=0$

Computing $D(n,m)$

- Imagine how best sequence handles the last characters $a_n$ and $b_m$
- If best sequence of operations
  - deletes $a_n$ then $D(n,m)=D(n-1,m)+1$
  - inserts $b_m$ then $D(n,m)=D(n,m-1)+1$
  - replaces $a_n$ by $b_m$ then $D(n,m)=D(n-1,m-1)+1$
  - matches $a_n$ and $b_m$ then $D(n,m)=D(n-1,m-1)$

Sequence Comparison: Edit Distance

- Given:
  - Two strings of characters $A=a_1 \ a_2 \ ... \ \ a_n$ and $B=b_1 \ b_2 \ ... \ b_m$
- Find:
  - The minimum number of edit steps needed to transform A into B where an edit can be:
    - insert a single character
    - delete a single character
    - substitute one character by another
Recursive algorithm $D(n, m)$

- if $n=0$ then
  - return $(m)$
- elseif $m=0$ then
  - return $(n)$
- else
  - if $a_n = b_m$ then
    - replace-cost = 0
  - else
    - replace-cost = 1
  - endif
  - return min ($D(n-1, m) + 1$, $D(n, m-1) + 1$, $D(n-1, m-1) + \text{replace-cost}$)

Dynamic Programming

for $j = 0$ to $m$; $D(0, j)$ ← $j$; endfor
for $i = 1$ to $n$; $D(i, 0)$ ← $i$; endfor
for $i = 1$ to $n$
  for $j = 1$ to $m$
    if $a_i = b_j$ then
      replace-cost ← 0
    else
      replace-cost ← 1
    endif
    $D(i, j)$ ← min ($D(i-1, j) + 1$, $D(i, j-1) + 1$, $D(i-1, j-1) + \text{replace-cost}$)
  endfor
endfor
### Example run with AGACATTG and GAGTTA

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Example run with AGACATTG and GAGTTA

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Reading off the operations

- Follow the sequence and use each color of arrow to tell you what operation was performed.