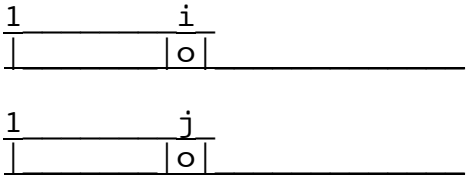
3-2-2010

Inexact Matches

ABBA -> ABBA (exact)
ABA -> AB-A (insertion/deletion)
ABCA -> ABCA (substitution)
ACA -> A-CA (insertion/deletion and substitution

S1 – S2
Edit (Levenshtein) Distance: What is the min # of edits (insertion, deletions, substitutions) made to S1 to change to S2?

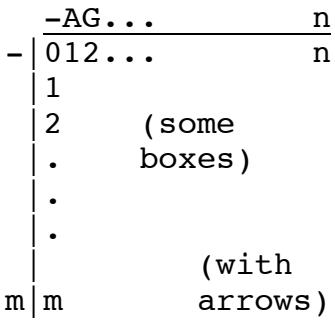
Example:
ABCA ---> ABCA
ACA ---> A-CA <- this is the correct alignment
ACA -x-> AC-A



E is edit distance between prefixes

E[i,j] = min:
-> E[i-1,j-1] + 1 if S1[i] != S2[j]
              + 0 otherwise
-> E[i,j-1] + 1
-> E[i-1,j] + 1

Fill dynamic programming array:



There's a much prettier picture here:
http://lslwww.epfl.ch/biowall/VersionE/ApplicationsE/SequenceE.html

List of Important Concepts(? I was distracted making that table when he was talking about this)

Global Alignment

Local Alignment – Did substrings in S1 & S2 that have the lowest edit distance

Gap Penalties – Pay for gaps as a block

3-4-2010

Kun-Mao Chao, William R. Pearson, Webb Miller. Aligning two sequences within a specified diagonal band. Bioinformatics 8(5):481-487.
bioinformatics.oxfordjournals.org/cgi/reprint/8/5/481

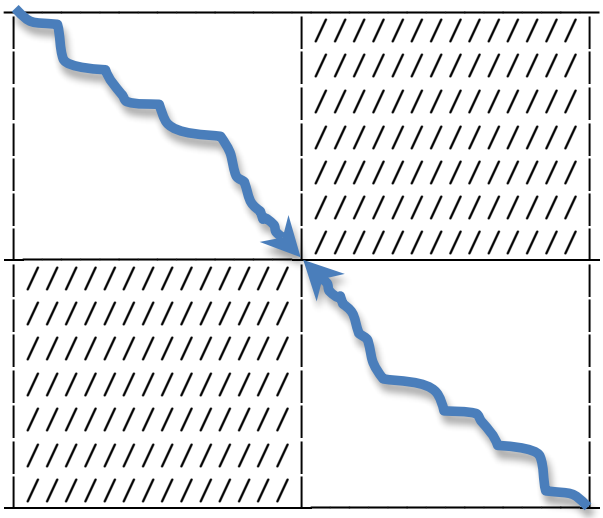
- This paper covers most of the material being covered on the midterm. **PRINT THIS OUT!**

```
ACTAA-CT
|  || ||
AG-AATCT
```

3 cases for variation:
-Insertion
-Deletion
-Substitution

```
E[i,j] = min{ E[i-1,j-1] +{ 1 ← S1[i] != S2[j]
                           { 0 ← S1[i] == S2[j]
              { E[i,j-1]+1
              { E[i-1,j]+1
```

(insert dynamic programming array and brief review)



- Determine middle row
- Compute score to that box starting from both the top left and bottom right
- Identify box with lowest combined score
- Using that as 2-way midpoint, divide the array into quadrants
- Discard the upper right, and lower left quadrants

- Repeat the algorithms on the upper left, and lower right quadrants

Run Time:
- n^2 for the first round
- $n^2/2$ for the second round
- etc…
- Approaches $2n^2$ run time

To account for runs of gaps:
g(n) = cost of n gaps in a row
g(n) = f(g(n-1))
$g(n) = g_o + g_e n$
g_o = cost of opening a gap
g_e = cost of extending a gap

```
E[i,j] = min{ E[i-1,j-1] +{ 1 ← S1[i] != S2[j]
                           { 0 ← S1[i] == S2[j]
            { min k<j → E[i,j-k]+g(k)
            { E[i-1,j]+1
```

Improvement → Store scores in 3 different tables:
E → scores of alignments ending in gaps in S1
F → scores of alignments ending in gaps in S2
G → scores of alignments ending with aligned characters

V = min(E,F,G) → score of alignment

```
E[i,j] = { E[i,j-1] + g_e
         { V[i,j-1] + g_e + g_o
```