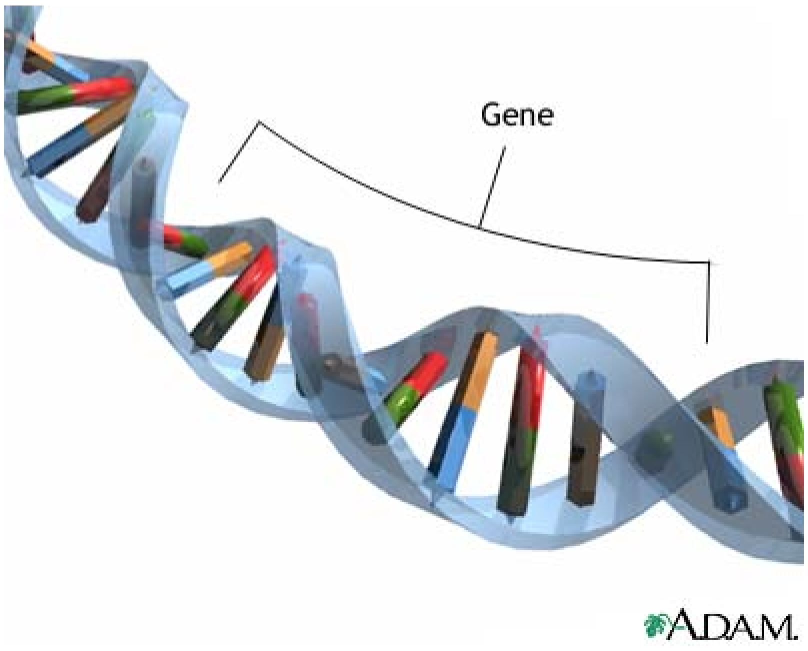
Android Application: Pairwise Gene Sequence Alignment

Zach Romer
Paul Stasiuk

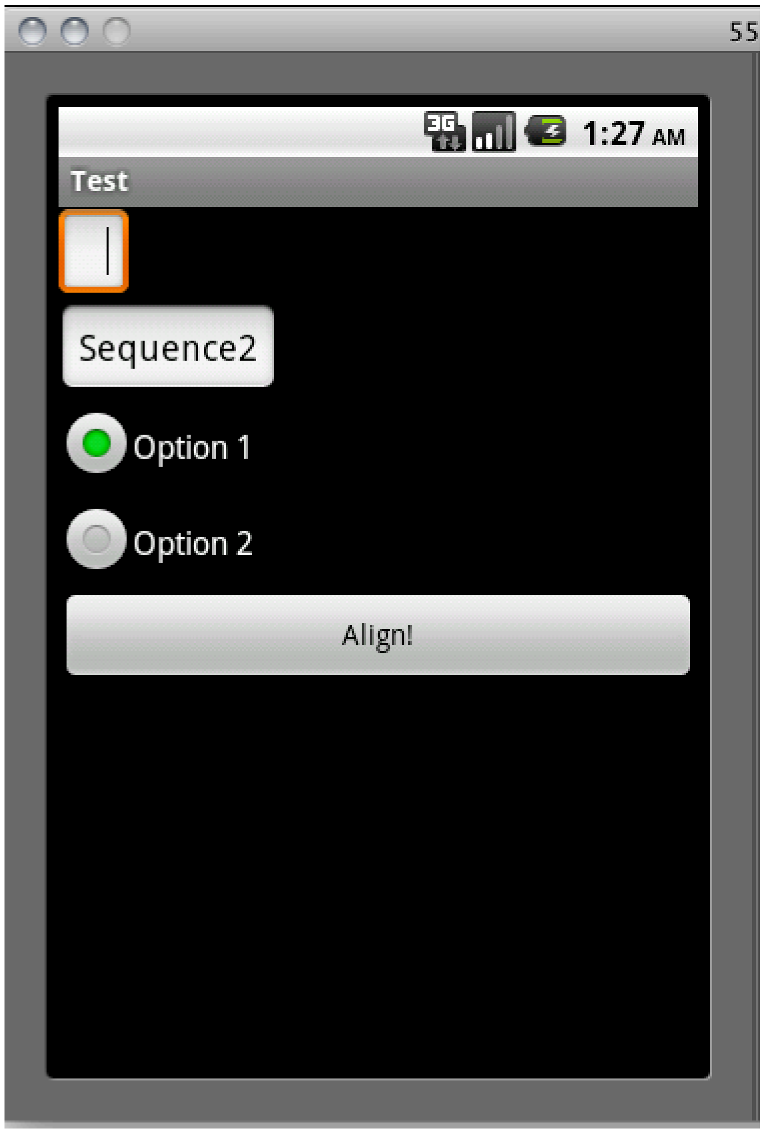
Abstract:

Pairwise gene sequence alignment is a long and tedious process to complete by hand. It involves graphing two sequences in a matrix style layout. With longer sets of letters, this could take longer amounts of time. The Pairwise Alignment Android application take the tedious process of graphing the letters by hand and lets powerful Android devices do all the calculating for the user. Implementing the Needleman-Wunsch and Smith-Waterman algorithms, this application allows those with an interest in bio-informatics input, align, and graph sequences of virtually any length(limited). The text boxes auto wrap to the length of the input and with a simple click of a button the application displays the aligned sequence.

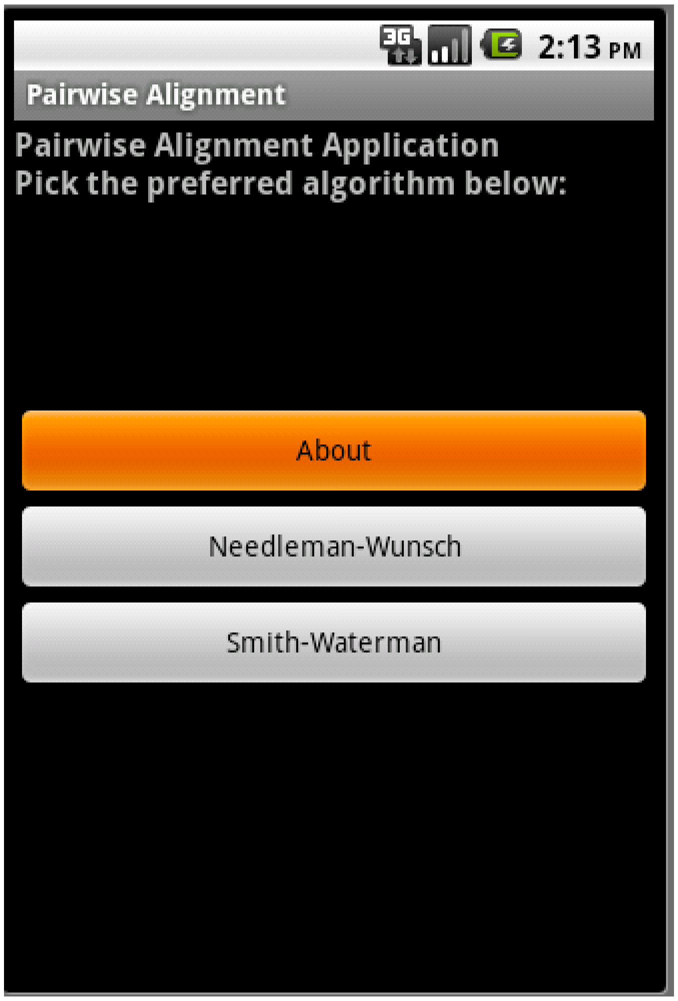


Project Description:

This application originally began as a single-view with radial buttons allowing the user to select his/her algorithm preferences. It was quickly realized that a single view was not only rudimentary, but also lacked functionality, as well as expandability for future additions.

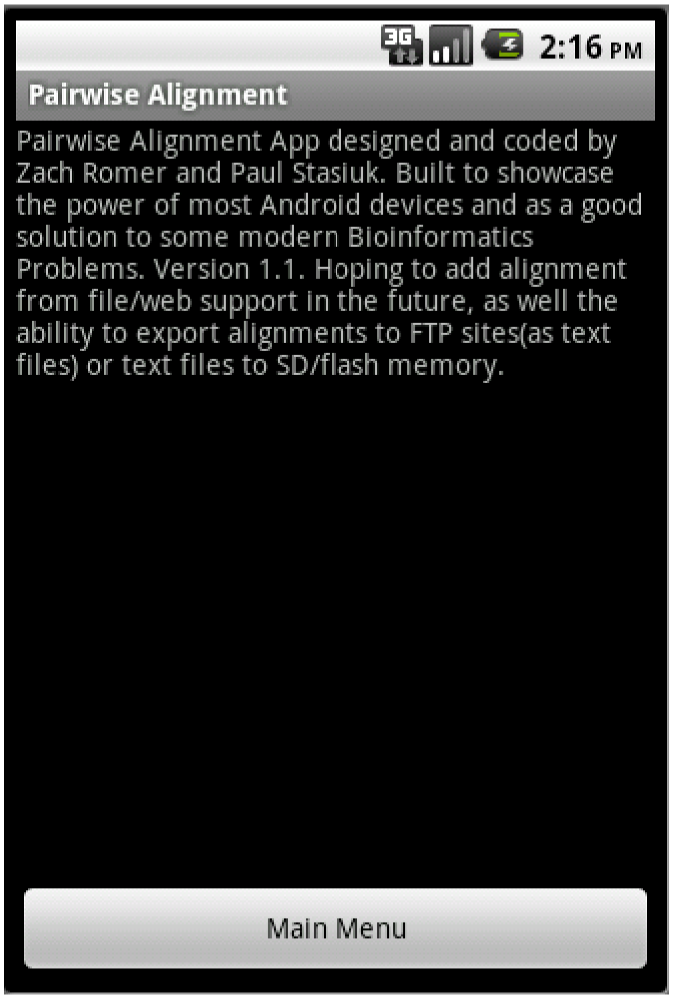


The updated main screen greets the user with a simple, three button menu which then branches out to respective pairwise alignment algorithm.



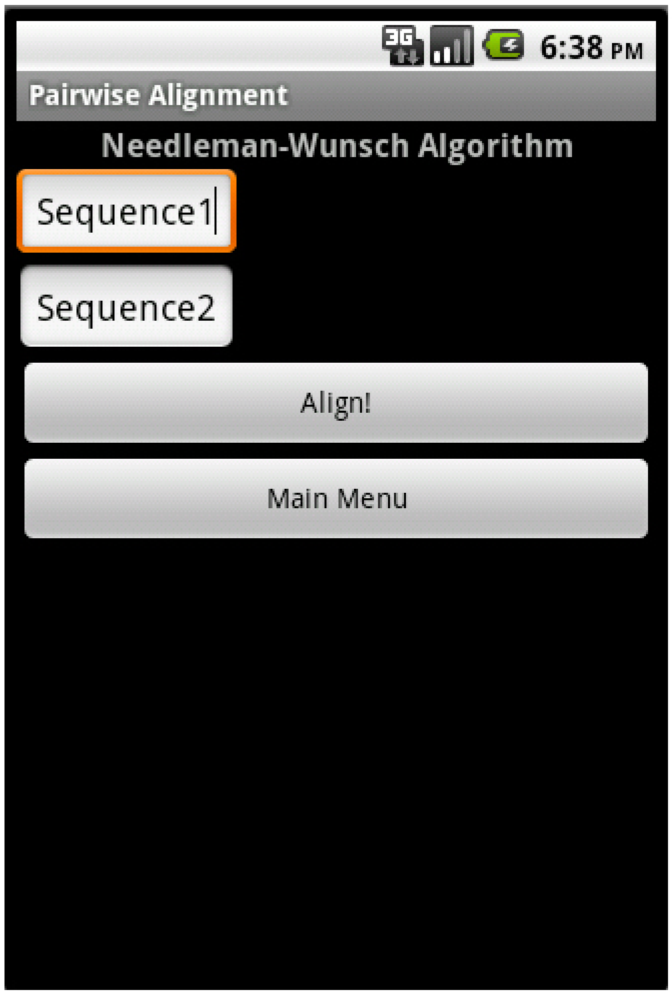
Pairwise Alignment Application
Pick the preferred algorithm below:

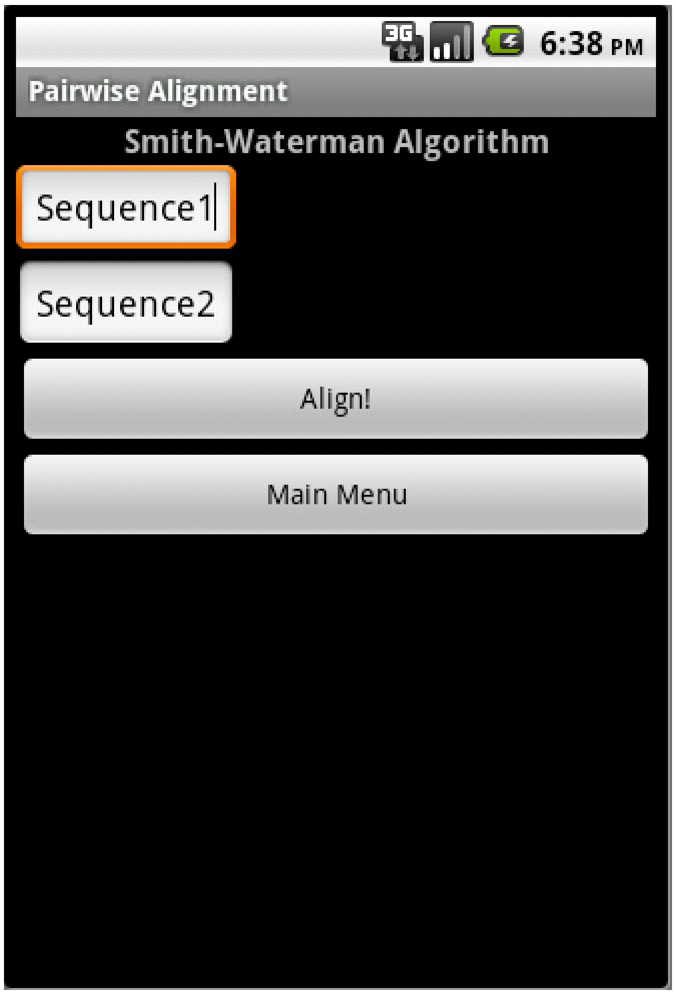
The “About” button takes the user to an “About Us” screen which gives some information about Zach and Paul, the “version” of the application, its purpose and future plans:



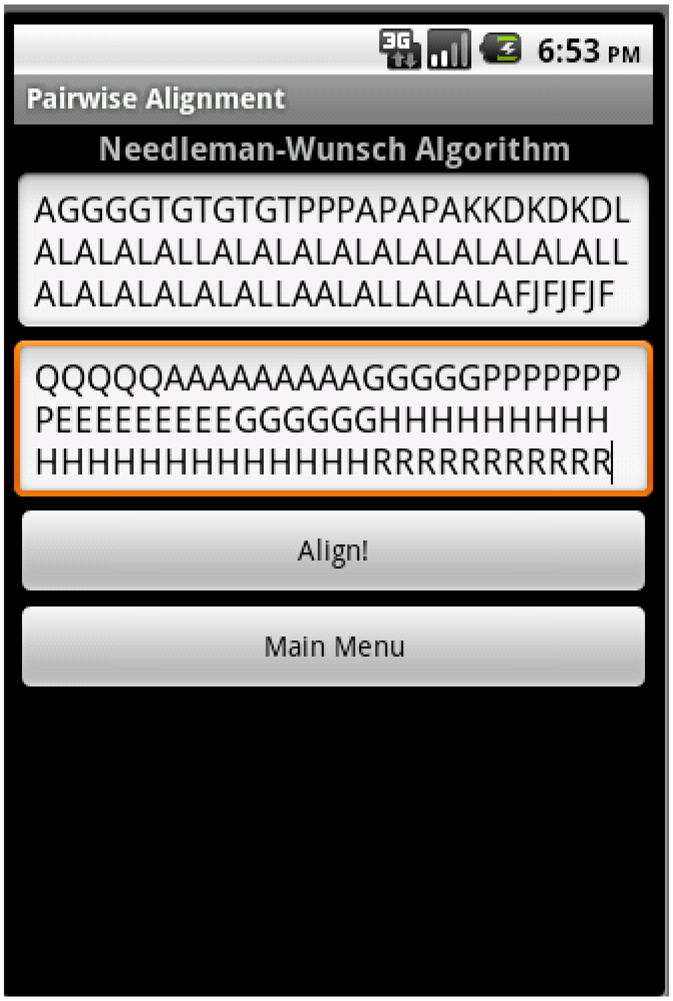
Pairwise Alignment App designed and coded by Zach Romer and Paul Stasiuk. Built to showcase the power of most Android devices and as a good solution to some modern Bioinformatics Problems. Version 1.1. Hoping to add alignment from file/web support in the future, as well the ability to export alignments to FTP sites(as text files) or text files to SD/flash memory.

The Needleman-Wunsch and Smith-Waterman buttons take the user to two different algorithms. This process allowed us to switch activities which allowed us to separate the respective algorithms code.





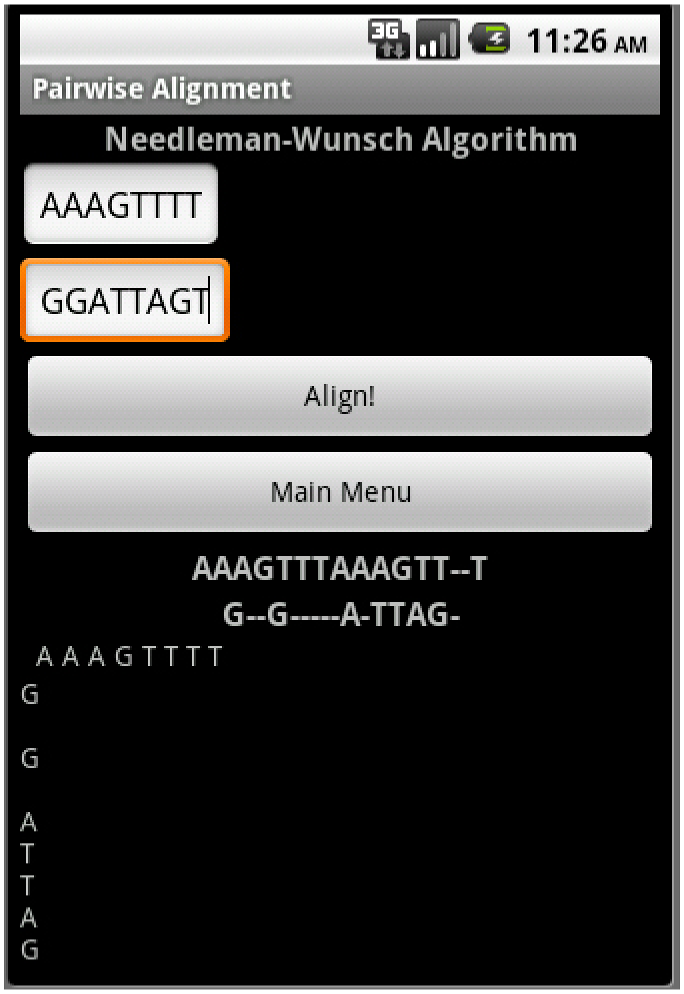
The text fields that contain Sequence1 and Sequence2 expand to fit a string of text the user wishes to input. Though it should be noted that to much text will eventually cut off the output(discussed in the Design section).



AGGGGTGTGTGTPPPAPAPAKKDKDKDL
ALALALALLALALALALALALALALALL
ALALALALALALLAALALLALALAFJFJFJF

QQQQQAAAAAAAAAGGGGGPPPPPPP
PEEEEEEEEEGGGGGGHHHHHHHHH
HHHHHHHHHHHHHRRRRRRRRRR

The code for the Pairwise Alignment was originally written as a simple Java program. This allowed us to compare outputs from our Android application to the outputs from the Java program. Once the user enters a sequence, they are presented with the aligned sequences, as well as a short visual representation on how the sequences were aligned(at the time of this writing, the matrix was not able to be formatted to display in the space below the sequences), sequences were run through both the Java application and the Android application and got the same results, ensuring consistency(note that the “dashes” are different lengths in the output from the Java application compared to the Android Application).

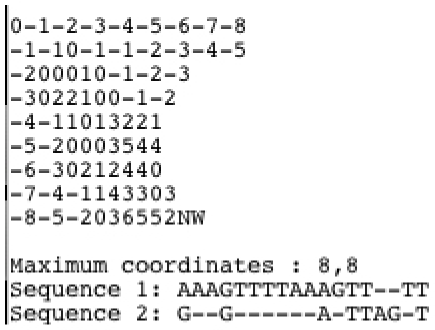


Needleman-Wunsch Algorithm

AAAGTTTT

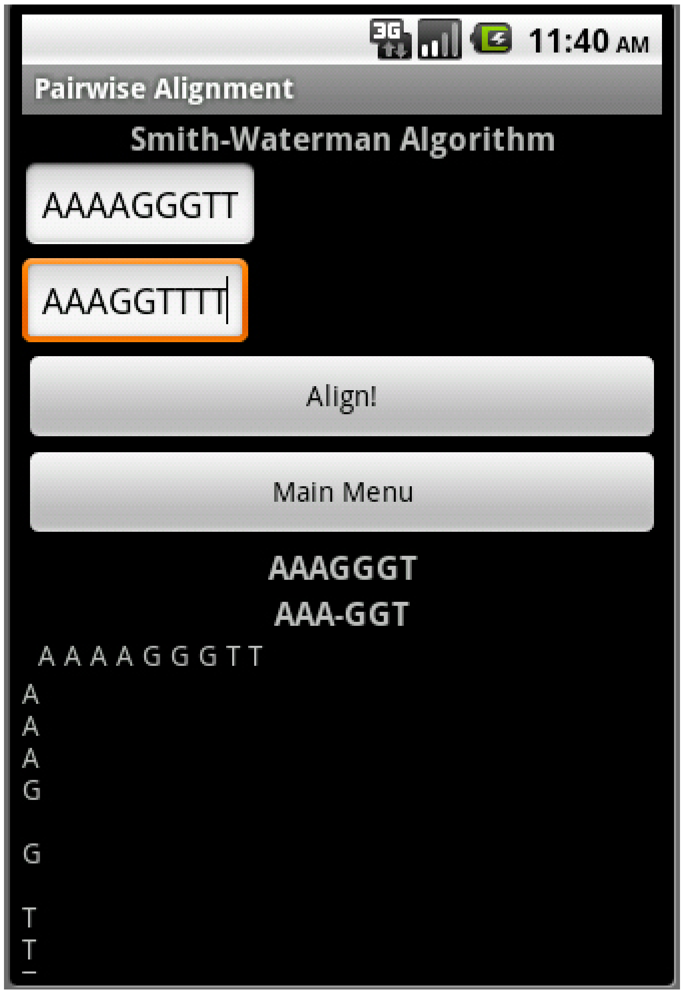
GGATTAGT

```
AAAGTTTAAAGTT--T
G--G-----A-TTAG-
 A A A G T T T T
G

G

A
T
T
A
G
```



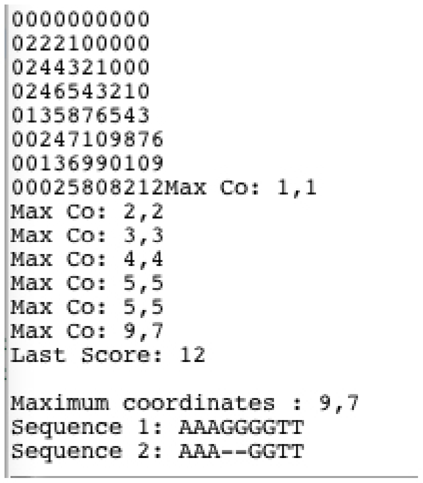
```
0-1-2-3-4-5-6-7-8
-1-10-1-1-2-3-4-5
-200010-1-2-3
-3022100-1-2
-4-11013221
-5-20003544
-6-30212440
-7-4-1143303
-8-5-2036552NW

Maximum coordinates : 8,8
Sequence 1: AAAGTTTTAAAGTT--TT
Sequence 2: G--G------A-TTAG-T
```

As seen about and below, the outputs are almost identical to those of the Java application, while the two algorithms display their respective alignments.



```
AAAAGGGTT
AAAGGTTTT
```

```
AAAGGGT
AAA-GGT
 A A A A G G G T T
A
A
A
G

G

T
T
```



```
0000000000
0222100000
0244321000
0246543210
0135876543
00247109876
00136990109
00025808212Max Co: 1,1
Max Co: 2,2
Max Co: 3,3
Max Co: 4,4
Max Co: 5,5
Max Co: 5,5
Max Co: 9,7
Last Score: 12

Maximum coordinates : 9,7
Sequence 1: AAAGGGGTT
Sequence 2: AAA--GGTT
```

Design(Java):

This program was done with more attention paid to simplicity of code and accuracy of results than to improved efficiency. The problem being approached requires the creation and analysis of a matrix of size n*m, where n and m are the number of elements in the two inputted sequences. This gives an amortized run time of n^2, which is not fast for big n; seeing as nucleotide sequences used by biologists are often in excess of 10000 nucleotides, this particular algorithm has limited applicability, especially when differing alignments are possible

The basic structure of the program(java version) consists of two levels: the MatrixNode class, and the MatrixNW class. The MatrixNode class is used to create objects which hold a score, path pointers showing where the score came from, and a string used to identify whether the particular node resulted from a Gap, Match, or Mismatch. MatrixNode contains methods for returning each of these values individually, and one method for setting all of the values contained in MatrixNode;

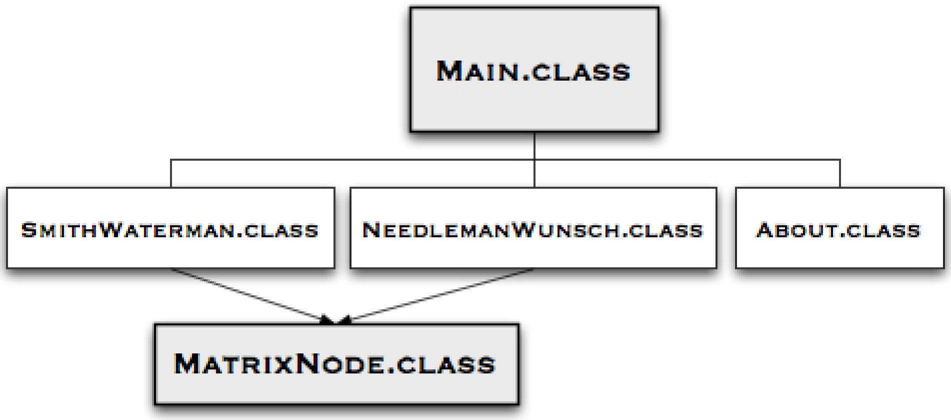
The MatrixNW constructor creates a 2D array, sized by the lengths of the input sequences (+1 to allow for initialization column and row) which holds MatrixNodes at each index. The parameter list for instantiating MatrixNW includes a boolean which determines whether the matrix will be scored as a Needleman-Wunch matrix or a Smith-Watermann matrix. The .fillMatrix(int mismatchPenalty, int gapPenalty, int matchScore) first initializes the first column and row with scores based on if the user wants a Smith-Watermann matrix or Needleman-Wunch. It then calculates the possible scores for each node in the matrix, and chooses the largest to set as the score for that node. If the matrix is called as a Smith-Watermann, the scores are evaluated before being set to the node; if they are less than 0, the score is set to 0 (Smith-Watermann does not allow scores less than 0). The pathX, pathY, and outcome variables are also set here for use in the .findPath() method.

The .findPath() method in MatrixNW traces a path back from the end of the matrix (or maximum value if the matrix is called as a Smith-Watermann), comparing the pathX and pathY values to the current position in order to find the next node in the sequence. Each time a node is accessed, a character is accessed from the corresponding spot in the appropriate input sequence and added to its respective alignment, represented at this point as a stack. Stacks provided the perfect FILO functionality for this; the .findPath() method begins building the sequences from the end (reverse order), so when the characters are popped from the aligned stacks they are returned in the correct order. A conditional checking for isSW is included in .findPath(), which exits the for/each alignment loop as soon as a node score = 0 (Smith-Watermann alignment is local, and only aligns parts of segments with scores > 0). Stacks also resize automatically, so needing to know what size the aligned sequences will be is avoided.

MatrixNW also contains a method returning the matrix as a 2D array of type MatrixNode, a method used to print the matrix, and a method to return the final aligned sequences.

Design(Android):

Moving the program to android proved to be more difficult than originally thought. The user interface(UI) design was originally structured as a single view, but soon became a main view that branched off into three other views. The way android handles moving Objects(or any type of data) between classes influenced the “three-branch” approach that was taken. Each “branch” extended the Android Activity class to provide Android functionality. The Main Activity is the Activity that provides the user with the ability to switch to an About Us page, Smith-Waterman algorithm, and NeedleMan-Wuncsh algorithm. The SmithWaterman and NeedleMan Activity both utilize the features of the standard MatrixNode.class(as written about above).



A challenging aspect of developing this application for Android was handeling input and output text. Because of the difficulties encountered, the NeedlemanWunsch Activity and SmiithWaterman Activity both contain the same code that is contained in the MatrixNW class(as written about above), only modified to accept inputs from Android and display outputs. This was done at the sacrifice of program size and potential for inefficiency because there is almost identical code in both activities save for a simple Boolean value altered to determine whether the code was outputting for the Smith-Waterman algorithm or for the Needlman-Wunsch algorithm. As of this writing, the formatting and exportation of the actual matrix that the alignment matrix does not display properly because of space constraints.

Sources:

1. Eclipse with Google Android SDK installed for Android application development
    a. http://developer.android.com/sdk/index.html
2. Google Dev Docs referenced for most of the code written
    a. http://developer.android.com/guide/appendix/faq/commontasks.html
    b. Specifically the UI/View Section:
        i. http://developer.android.com/guide/topics/ui/index.html
3. MobiForge Tutorials referenced for most button tutorials and layout
    a. http://mobiforge.com/designing/story/understanding-user-interface-android-part-1-layouts
    b. http://mobiforge.com/designing/story/understanding-user-interface-android-part-2-views
    c. http://mobiforge.com/designing/story/understanding-user-interface-android-part-3-more-views
    d. http://mobiforge.com/designing/story/understanding-user-interface-android-part-4-even-more-views
4. BlueJ for Java application development
    a. http://www.bluej.org/

5. String to Character Array Code snippets(modified to work with android)
    a. http://www.javadb.com/character-array-to-string-conversion

```
  public void convertCharArrayToString() {
     char[] charArray = new char[] {'a', 'b', 'c'};
     String str = new String(charArray);
     System.out.println(str);
 }
```

6. Google Android API
    a. http://developer.android.com/reference/packages.html
    b. Specifically the VIEW sections
7. Pairwise Gene Sequence Alignment for development of algorithms and techniques
    a. http://docencia.ac.upc.edu/master/AMPP/slides/ampp_sw_presentation.pdf
    b. Pairwise Alignment PPT- Loyola University Chicago
8. Image on first page:
    a. http://www.topnews.in/health/files/Genes.jpg
9. Graphs/Matrix image made with OmniGraffle Pro
    a. http://www.omnigroup.com/products/omnigraffle/