Daniel Green
Grayland Lunn
Alejandro Tovar
2/21/19

CS325 Assignment 2

Algorithm Pseudo Code

The algorithm to generate the alignment matrix as follows:
Add an “edit” to the first position of each word A and B.
Add the edit cost for the padding characters in the position j=0 or i=0

Iterate over the entire edit distance matrix (for i in [0 ... len(A)]) (for j in [0 … len(B)]), add the values E[i][j] to the matrix that correspond to the minimum value of (cost of an edit to A[i], cost of an edit to B[j], cost of A[i] and B[j] being paired and not edited). Also, store which direction the edit came from.

Build the aligned strings from the alignment matrix using the edit direction matrix, and find the optimal substring to find the shortest path from index[i][j] to index [0][0]. This path is then the directions for creating the alignment.

The algorithm that edits the string’s alignment as follows:
Create an empty string to put the edited string in

If the string is the first string, then loop through the list of directions. If the direction is ‘left’, insert a space. If the direction is ‘up’, then do nothing and add the current letter. If the direction is ‘diagonal’, do nothing and add the current letter.

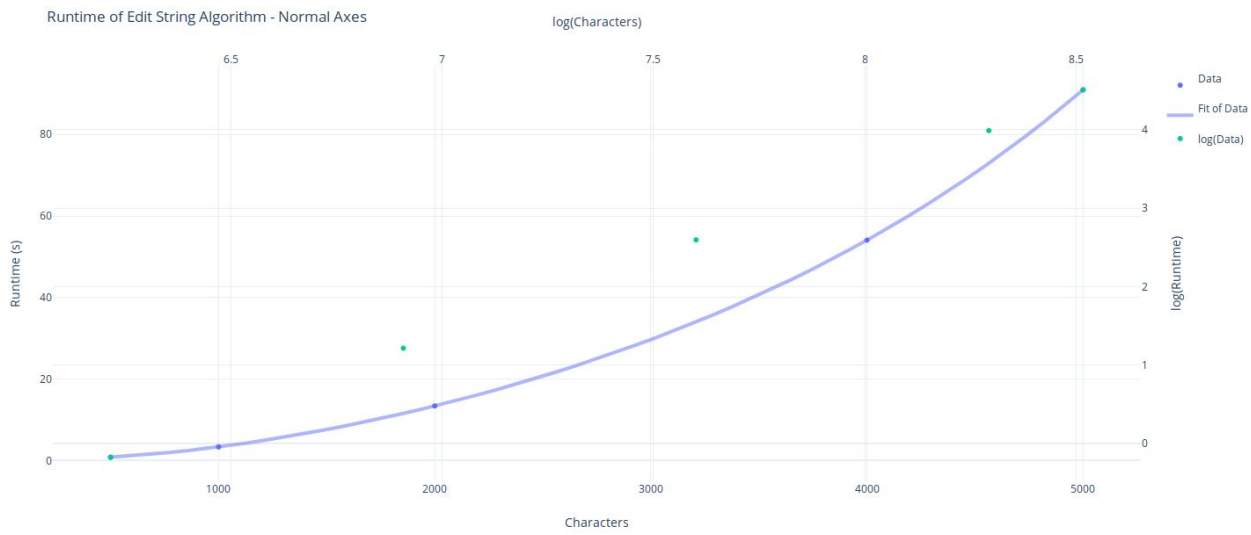
Else then it is the second string, then loop through the list of directions. If the direction is ‘left’, then do nothing and add the current letter. If the direction is ‘up’, insert a space. If the direction is ‘diagonal’, do nothing and add the current letter.

When the loop ends return the new aligned string.

Assymptotic Analysis of Runtime

For two words of length n and m, and n*m matrix is generated using the characters of the two words. A constant number of operations are done for each entry into the matrix, so the runtime is c*n*m. If both the words are the same length, the runtime simplifies from $\Theta(mn)$ to $\Theta(n^2)$ with the given words because n = m.

Reporting and Plotting of Runtime



A look at the runtime shows the log log plot to have a slope of 2.023, very close to the expected 2. The runtime is collected by comparing the start and end time for the edit cost matrix generation function, and then averaged over the course of 10 calls to the generator. This runtime is only for the generation of the matrix, not for the generation of the output words or any other operation, so that the runtime considers as few variables as possible.

Runtimes were generated on a Lenovo W450-S laptop with a 2.4GHz Intel i7-5500 processor. These runtimes are likely slower than those you would expect with a server or using a faster (compiled) programming language.

Here are the data points we collected:

Len	Runtime (s), *average of 10 executions*
500	0.838345170021
1000	3.36984739304
2000	13.4107764006
4000	54.0775579214
5000	90.9415251493

The experimental running time of this algorithm is approximately $\Theta(n^{2.02})$, however more tests with data in between the measured points is recommended before a definitive value can be reached. The runtime for a single loop of the algorithm could vary by more than 2%, so 2.02 is well within reasonable error for 5 data points.

Interpretation and Discussion

The growth curve of the runtime plot very closely resembled the expected data. The log-log plot had a slope of 2 (within the relative error of the collected measurements) which is expected for an algorithm with runtime of $\Theta(n^2)$ and the plot of the actual points looks very close to what you

would expect. Doubling the number of characters being edited quadruples the runtime almost exactly, so $\Theta(n^2)$ is very closely reflected in the results.