Project 2 Report

Kenneth Sills, Kevin Orr, Elijah Malaby

July 15, 2017

1 Breaking Down Problems

We separate the transpositions from all other typo forms. By first keeping a record of where possible transpositions can occur (and each permutation of possible propagation lengths), we vastly simplify the recurrence. In the memoized solution, we also separate the initial table fill from backtracking.

2 Parameters for the recursion

The two arrays p and t (pattern and typo) and indexes i and j representing the positions in p and t currently being compared

3 What recurrence can you use

4 What are the base cases

Whenever i or j are equal to 1. If $i = 1$, the remaining characters in $t[1..j-1]$ were trivially all insertions at the beginning. If $j = 1$, the remaining characters in $p[1..i-1]$ were deleted. If both $i = 1$ and $j = 1$, there are no further characters to compare.

5 What data structure would you use

A map from pairs of (i, j) to the cost of the recurrence for (i, j), as well as a map from pairs of (i, j) to a set of numbers of transpositions k such that for each a chain of transpositions may have occurred in $p[(i-k)..i]$ and $t[(j-k)..j]$.

We found that the best structure to use for memoizing would be 2D array with each cell containing the lowest cost typo, the overall cost of accumulated typos, and the index to the lowest cost next cell.

As well, a hash table is kept, indexed by the squashed coordinates of our 2D array to store possible transposition sites. This hash table contains arrays of size 12, where the distance into the table is the number of propagating transpositions, the contents of which is the cost to perform that chain of transpositions.

6 Pseudocode for a memoized dynamic programming solution

7 Worst case time complexity

The worst-case of our algorithm will encompass:

- O(n * m) Transposition identification.
- O(n * m) Fill table pass
- O(n + m) Backtracking pass

So, as a whole, the algorithm takes O(n * m) time complexity.

```
   Input: data: Table containing the memoized data
   Input: transposes: List of possible transpositions
   Input: correct: The correct string
   Input: actual: The actual string with typos
   Input: i: Current position into the correct string
   Input: j: Current position into the actual string
   Output: Running cost of typos
1  Algorithm Fill(data,transposes, correct, actual, i, j):
2  end
```

8 Pseudocode for nested loop

9 Can the space complexity of the iterative algorithm be improved relative to the memoized algorithm

10 Describe one advantage and disadvantage of the iterative algorithm