Joe Kuczek 4/22/2017
CS 311 PA #3

Report: Dynamic Programming and MinCostVC

- **minCostVC(int[][] M):**
  - Say we have n rows and m columns
  - Recurrence Relation:
    - We need to compute the minimum direction of three different directions (down-left, down, down-right). Using recursion without memoization, this will be worst case 3^n since each recursive call can branch off three directions.
    - Relation: $T(n) = 3 * T(n) \in O(3^n)$
    - It may be possible to improve the runtime of this algorithm by caching a minVC path each time a path is found. Then, if we reach a point that has already been traversed by a path, we can just add all the new points from the previously traversed path to the current path giving us a new minVC.
  - Iterative Runtime:
    - The runtime will be $O(n * m)$ in nearly all cases.
    - There are two loops, one over columns for selecting a starting column and one over rows for traversing downwards selecting a minimum cut.
    - For each point, the minimum direction is stored (left, middle, right) so we don't need to compute those multiple times.
- **stringAlignment(String x, String y):**
  - Say length of x = n and length of y = m
  - Recurrence Relation:
    - We need to compute three different costs for each position in Strings x and y. If we do this using recursion, it would take 3^n time since each character index is divided into three different problems. The three problems are as follows:
      - Compute cost of not changing anything.
      - Compute cost of inserting '$' to the right of current symbol.
      - Compute cost of inserting '$' to the left of current symbol.
    - Relation: $T(n) = 3 * T(n) \in O(3^n)$
    - Note we aren't performing divide and conquer, each subproblem is precisely the same size as the original base case.
  - Iterative Runtime:
    - The runtime of this algorithm will be $O(n * m)$ in nearly all cases.
    - There are two loops, one loop over String x and one over String y.
    - The inner loop computes costs for three different options. These must be considered at every index position (i, j) where i and j are character indices for String x and String y respectively.
    - The reconstructPath() method will traverse the newly created minimum penalty path. This should take $O(1)$ since for every cell we also stored the parent cell. Note that I had problems in the reconstructPath() method, I know I computed the optimal matrix but my path is incorrect.