Jeff Hildebrandt

COT5405

Programming Assignment 3

**Introduction**

This assignment’s purpose is to demonstrate how dynamic programming can be used in several different real-world examples. This will be demonstrated by predicting RNA secondary structures, calculating the sequence alignment of two strings, and finding the max-flow of a directed graph.

**Method**

Language:

Python 3.x

Compilation instructions:

Problem 1:

1. Cd into folder Prob1
2. Run the command: python3 ./RNASecondaryStructure.py

Problem 2:

1. Cd into folder Prob2
2. Run the command: python3 ./SequenceAlignment.py

Problem 3:

1. Cd into folder Prob3
2. Run the command: python3 ./MaxFlow.py

Format of Input

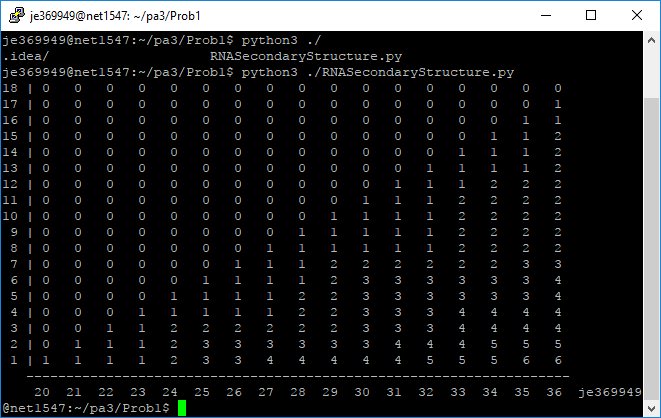
The input for all the problems is read in as a hard-coded array.

**Problems**

1.

This program predicts the RNA secondary structure for a given RNA sequence. The RNA sequence is read in as an array of characters where each character is a new index of the array. It follows the rules for no sharp turns, and no crossing. It prints out the finished matrix as demonstrated on slide 38 of the dynamic-programming.ppt. The running time is the same as shown in the slides which is O(n^3).

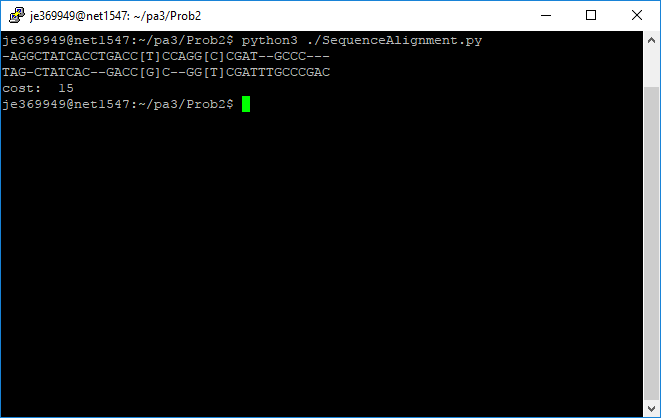
A.



B. I was able to recreate the output as defined on slide 39 in dynamic-programming.ppt, unfortunately I was unable to figure out how to trace through this output to create the RNA secondary structure.

2.

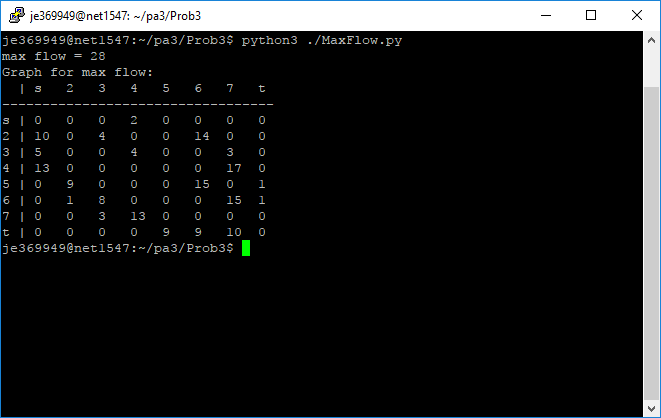
This problem shows sequence alignment on two strings. It uses a mismatch penalty of 2 and a gap penalty of 1. The algorithm is done the same as demonstrated in the slides. The only difference is I went backwards to populate the array instead of forwards. It accepts two strings as input where each string is an array where each character is an index in the array. The time complexity is O(mn) where m is the length of the first string and n is the length of the second string. Both a and b are shown in the output window.



3.

This problem finds the max flow of a directed graph. It accepts the directed graph input as a 2D array represented as an adjacency matrix. It prints out the max flow followed by the end directed graph as an adjacency matrix. The algorithm runs in O(max flow \* number of possible paths from s to t).

A.



B.

