CS5800 Algorithms Spring 2015 Assignment #6

1. (30 points) We wish to find the score of the best alignment between the following two RNA sequences: UAAGGUGCAUCUAGUGCUGUUAG and UAAGUGCGUGCAUGUAUAUGUG. Each match scores 2 points, each gap scores -2 points, and each mismatch scores -3 points. We wish to maximize the number of points. Show the matrix for the execution of the dynamic programming algorithm for this alignment problem, and show an optimal alignment. Note that in this problem we are maximizing a score rather than minimizing an edit distance.

These RNA sequences are the actual miRNA sequences mir-18b and mir-467c. One is human and the other mouse.

2. (25 points) Compute the all-pairs shortest paths for the following graph:

Edge	**Length**	**Edge**	**Length**	**Edge**	**Length**	**Edge**	**Length**	**Edge**	**Length**
(1,4)	-1	(1,5)	-2	(2,4)	3	(2,5)	5	(3,2)	4
(3,5)	5	(4,2)	3	(4,3)	4	(5,2)	4	(5,3)	4

Show your answer as a sequence of 5x5 matrices. You do not have to show the shortest paths.

3. (30 points)

Blocks World. You have a collection of rectangular blocks. For example, some 1x1x1 blocks, some 1x2x2 blocks and some 2x2x2 blocks. You want to stack the blocks so that they are as high as possible. A block can be stacked on another one only if its bottom is strictly smaller than the top of the block on which it is stacked. For example, one can stack a 1x1x1 block on top of a 1x2x2 block or a 2x2x2 block a 2x2 face on the top (and bottom). However, it is not possible to stack either a 1x2x2 block or a 2x2x2 block on top of the other one. So the maximum height in this case is 3.

Prove that in a stack there can be at most two blocks with the same shape.

Assume that for each shape one has exactly two blocks. Develop a polynomial-time algorithm for determining the maximum height of a stack. Compute and prove the time and space complexity of your algorithm.

4. (15 points) Find the longest strictly increasing subsequence in the following sequence:

```
8, 17, 9, 4, 14, 19, 8, 1, 15, 9, 17, 4, 4, 10, 5, 8, 14, 19, 12, 14
```

Show each step of the algorithm. For each step show the length as well as the position of the previous entry.

Show the first longest strictly increasing subsequence that was found. Also show the positions of the entries in the longest strictly increasing subsequence.