CPSC 320 2022W1: Take-home Test 2 Solutions

1. (0 points) I hereby pledge that I have read and will abide by the rules, regulations, and expectations set out in the Academic Calendar, with particular attention paid to:

   - The Student Declaration
   - The Academic Honesty and Standards
   - The Student Conduct During Examinations
   - And any special rules for conduct as set out by the examiner.

   I affirm that I will not give or receive any unauthorized help on this examination, that all work will be my own, and that I will abide by any special rules for conduct set out by the examiner.

   ○ True ○ False

2. (5 points) True or False?

   (a) (1 point) NP-Complete problems can be solved (exactly) in exponential time.

   ● True ○ False

   (b) (1 point) The longest common subsequence problem from worksheet 8 belongs to NP.

   ● True ○ False

   (c) (1 point) If we proved that the *minimum weight point set triangulation* problem is NP-Complete, then we would have shown that there is no polynomial time algorithm that can solve it.

   ○ True ● False

   (d) (1 point) Part of an NP-completeness proof for a new problem $\mathcal{P}$ involves showing how to reduce $\mathcal{P}$ to a known NP-Complete problem in polynomial time.

   ○ True ● False

   (e) (1 point) If we could solve the *Minimum Test Collection* problem (a known NP-Complete problem) in polynomial time, then we could also solve the *Minimum Dominating Set* problem (a problem that belongs to NP) in polynomial time.

   ● True ○ False

3. (15 points) Consider the following problem: we are given a matrix $M[1 \dots m][1 \dots n]$ of integer values, and we want to find the path from an entry in the first column to an entry in the last column with the largest sum, subject to the constraint that the path can only contain one entry per column (so it must go from left to right, with no doubling-back allowed).

   Here are two examples with circles around the entries of the path with the largest sum.

$$\begin{pmatrix} 2 & 6 & 4 & 5 \\ 3 & 9 & 1 & 6 \\ 4 & 8 & 2 & 7 \\ 1 & 2 & 3 & 9 \end{pmatrix} \qquad \begin{pmatrix} 6 & 2 & 3 & 7 \\ 3 & 3 & 1 & 5 \\ 8 & 4 & 3 & 1 \end{pmatrix}$$

   In the first example, the optimal path (shown in circles) is $M[3][1]$, $M[2][2]$, $M[3][3]$, $M[4][4]$, while in the second example it is $M[3][1]$, $M[2][2]$, $M[1][3]$, $M[1][4]$.

   (a) (2 points) A greedy algorithm that finds the path with the largest sum by starting with the largest element in column 1, and picking at each step the largest adjacent element in the next column, will find the path with the largest possible sum:

   ○ Always ● Sometimes ○ Never

(b) (4 points) Let $S[i][j]$ represent the maximum possible sum of a path that starts in column 1 and ends at $M[i][j]$. Complete the following recurrence relation for $S[i][j]$:

$$S[i,j] = \begin{cases} -\infty & \text{if } i < 1 \text{ or } i > m \\ M[i][j] & \text{if } 1 \leq i \leq m \text{ and } j \boxed{= 1} \\ \boxed{M[i][j] + \max\{S[i-1][j-1], S[i][j-1], S[i+1][j-1]\}} & \text{if } 1 \leq i \leq m \text{ and } j \boxed{> 1} \end{cases}$$

(c) (3 points) Which of the following (incomplete) iterative algorithm will compute the entries $S[i][j]$ correctly? Assume that an algorithm can only be incorrect because of an erroneous loop structure. If you were unable to come up with a plausible looking recurrence relation in part (a), use:

$$S[i,j] = \begin{cases} -\infty & \text{if } i < 1 \text{ or } i > m \\ M[i][j] & \text{if } 1 \leq i \leq m \text{ and } j \leq 3 \\ M[i][j] + \min\{M[i-1][j-1] + S[i-2][j-2], M[i+1][j-1] + S[i+2][j-2] & \text{if } 1 \leq i \leq m \text{ and } j > 3 \end{cases}$$

to answer this question.

Algorithm A:

```
// Assume the base cases have been taken care of.
for i ← 1 to m
  for j ← 1 to n
    S[i][j] ← ...
```

Algorithm B:

```
// Assume the base cases have been taken care of.
for j ← 1 to n
  for i ← 1 to m
    S[i][j] ← ...
```

- ○ Only algorithm A will work.
- ● Only algorithm B will work.
- ○ Both algorithm A and algorithm B will work.
- ○ Neither algorithm A not algorithm B will work.

(d) (5 points) Complete the algorithm below, that takes in the arrays M and S and returns the path with largest sum. Assume that the elements of M in row 0 and $m+1$ exist and contain the value $-\infty$. If you were unable to come up with a plausible looking recurrence relation in part (a), use:

$$S[i,j] = \begin{cases} -\infty & \text{if } i < 1 \text{ or } i > m \\ M[i][j] & \text{if } 1 \leq i \leq m \text{ and } j \leq 3 \\ M[i][j] + \min\{M[i-1][j-1] + S[i-2][j-2], M[i+1][j-1] + S[i+2][j-2] & \text{if } 1 \leq i \leq m \text{ and } j > 3 \end{cases}$$

to answer this question.

```
Algorithm returnPath((M[1... m][1... n], S[1... m][1... n])

  //
  // Find largest element in the last column.
  //
  k ← 0
  for i ← 1 to m do
    if S[i][n] > S[k][n] then
      k ← i

  // Fill in the rest.
```

```
solution ← empty list
for j ← n down to 1 do
  insert (k, j) in front of solution
  if S[k][j] = M[k][j] + S[k-1][j-1] then
    k ← k - 1
  else if S[k][j] = M[k][j] + S[k+1][j-1] then
    k ← k + 1
```

(e) (1 point) What is the space complexity of this algorithm?

Solution: It takes $\Theta(mn)$ space (the size of the array S).