COMP 3804 – Design and Analysis of Algorithms
Assignment 3

Due: March 17, 2017 at 23:55

- Your solutions should be submitted online on cuLearn in the form of a single PDF file.
- Your answers should be precise, concise and clear. All algorithms should be given in pseudocode.
- Every part of every theory question is worth 2 marks. The grading scheme is 2 points for a correct answer, 0 for a completely incorrect answer, and 1 point for something in-between. The implementation part (question 4g) is worth 5% of the grade.

1. A classmate suggests the following greedy strategy for the Matrix Chain Multiplication problem (see Lecture 10): always multiply the adjacent pair with the maximum shared dimension (break ties arbitrarily). For example, if we are given matrices A_1, A_2, and A_3, with dimensions 10×100, 100×5, and 5×50, this would suggest first multiplying A_1 and A_2, since their shared dimension is 100, while the shared dimension of A_2 and A_3 is only 5. This is the optimal solution for this particular example.

   Either prove that this always results in an optimal solution, or give a counter-example where there is a better multiplication order than the one given by this strategy.

2. Consider the following problem: given a set of intervals $S = \{(a_1, b_1), \ldots, (a_n, b_n)\}$, find the largest subset of *disjoint* intervals. Two intervals (a_1, b_1) and (a_2, b_2) are disjoint if they do not overlap other than at their endpoints; in other words, if $b_1 \leq a_2$ or $b_2 \leq a_1$. For example, if $S = \{(1, 5), (3, 6), (5, 7)\}$, then the largest subset of disjoint intervals is $\{(1, 5), (5, 7)\}$.

   While brainstorming, you and your friends come up with four greedy strategies for this problem. For each strategy, prove or disprove that it always produces an optimal solution. Assume that ties are broken arbitrarily.

   (a) Pick the shortest interval (minimum $b - a$). Remove all intervals that overlap this interval. Repeat.

   (b) Pick the interval with the minimum start (a-value). Remove all intervals that overlap this interval. Repeat.

   (c) Pick the interval with the minimum end (b-value). Remove all intervals that overlap this interval. Repeat.

   (d) Pick the interval that overlaps the fewest other intervals. Remove all intervals that overlap this interval. Repeat.

   Select one of the above strategies for which you were able to prove that it always produces an optimal solution.

   (e) Give a greedy algorithm for this problem, based on your selected strategy.

   (f) Analyze the running time of your algorithm.

3. Given a set of points in the plane, the *minimum spanning tree* (MST) is the shortest tree that connects all the points (it minimizes the total edge length). Consider the following divide and conquer strategy for this problem:
   - Sort the points by x-coordinate.
   - Recurse on the $\lceil n/2 \rceil$ leftmost points.
   - Recurse on the $\lfloor n/2 \rfloor$ rightmost points.
   - Add the shortest edge that connects a point on the left to a point on the right.

   (a) Prove that this always returns a spanning tree.

   (b) Prove or disprove that this always returns an MST.

4. The board game Less is played on an 8×8 board with randomly placed walls between squares. Each player aims to move their pieces to the opposite corner of the board before the others do the same.

   We will consider a generalized single-player version of the game, where the board has size $n \times n$. The board is specified by a positive integer n and two $(n-1) \times (n-1)$ boolean arrays `leftWall` and `bottomWall`; `leftWall`$[i][j]$ is true if there is a wall on the left edge of square (i, j) and false otherwise, `bottomWall` is analogous for walls on the bottom edge of the square. Square $(0, 0)$ corresponds to the bottom-left, and square $(n-1, n-1)$ to the top-right.

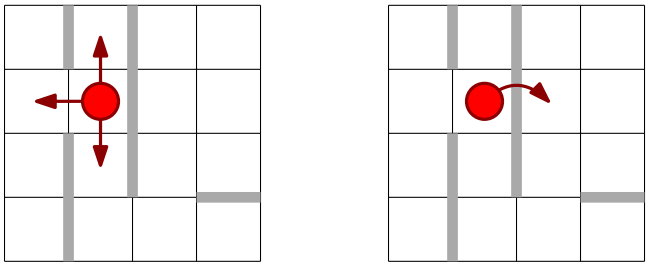

Figure 1: (left) Possible simple moves in the game of Less. (right) A wall jump.

   In one move, a player can move a piece one square horizontally or vertically if it is not blocked by a wall (see Figure 1, left). This is called a *simple move*.

   (a) Give an algorithm that uses breadth-first-search to find the minimum number of simple moves to move a piece from the bottom-left corner of the board to the top-right. If the top-right corner is unreachable, your algorithm should return ∞.

   (b) Why is your algorithm correct?

   (c) Analyze the running time of your algorithm.

   A player can also move a piece one square horizontally or vertically through a wall (see Figure 1, right). This is called a *wall jump*, and costs two moves.

   (d) Give an algorithm that finds the minimum number of moves required to move a piece from the bottom-left corner of the board to the top-right, using both simple moves (that count as 1 move) and wall jumps (that count as 2 moves). You may use any algorithm discussed in class as a black box.

   (e) Why is your algorithm correct?

   (f) Analyze the running time of your algorithm.

   (g) Implement your algorithm in the accompanying zip file.

   Every turn, each player gets three moves that they can use for any combination of simple moves or wall jumps. So during one turn, a player could perform three simple moves, a simple move followed by a wall jump, or a wall jump followed by a simple move. It is not necessary to use all three moves, so a turn could consist of just a single wall jump.

   (h) Give an algorithm that finds the minimum number of three-move turns required to move a piece from the bottom-left corner of the board to the top-right, using both simple moves (that count as 1 move) and wall jumps (that count as 2 moves). You may use any algorithm discussed in class as a black box.

   (i) Why is your algorithm correct?

   (j) Analyze the running time of your algorithm.