MCS 401 – Computer Algorithms I
Fall 2019
Problem Set 2

Lev Reyzin

Due: 9/30/19 by the beginning of class

Instructions: Atop your problem set, write your name and whether you are an undergraduate or graduate student. Problems labeled "(**U**)" and "(**G**)" are assigned to undergraduate and graduate students, respectively.

1. [10 pts] Consider a connected graph $G = (V, E)$, and a vertex $u \in V$. Let T_B be a BFS tree of G rooted at u, and let T_D be a DFS tree of G rooted at u. Prove that if $T_B = T_D$, then G is a tree.

2. [10 pts] Analyzing greedy algorithms:

(**U**) A large number of houses are being built on a straight street in a new development. As the city planner, you need to place fire hydrants such that each house is within 300 feet of one before anyone can move in. To save costs, you want to place as few fire hydrants as possible. Formalize this problem and devise an algorithm to optimally solve this problem. Prove your algorithm to be correct.

(**G**) In this Algorithms class, I want to cover topics $t_1, \ldots, t_n$. I should present all the n topics in that exact order because each topic t_i requires the students to have seen the material in topics t_1 to t_{i-1}. Each topic i takes no more than 50 minutes to present, and I have 50 minutes per lecture to cover as many topics as I'd like. I have also learned (the hard way) that I cannot split topics across lectures; otherwise students have a hard time following my lectures. Currently, I use the following simple strategy: on a given lecture, I cover as many topics as I can back-to-back, and I stop once the time remaining in the lecture is smaller than the amount of time I need to cover the next topic. Then, I let students ask me questions (or otherwise use the remaining time) or simply end class a little early.

However, I recently had an idea. What if I sometimes didn't fill up a lecture with topics as fully as possible, so that some of the later lectures would be better "packed"? My hope is that if I change my strategy, we might be able cover all the material for this class in fewer lectures. Then I could finish teaching early and fly off to Montana to do some hiking in Glacier National Park, which I hear is beautiful in Spring. Could my idea possibly work, or are we all stuck here until the end of the semester? Formalize this problem and then prove why or why not.

3. [10 pts] Suppose we are given an instance of the Shortest s–t Path Problem on a directed graph G. We assume that all edge costs are positive and distinct. Let P be a minimum-cost s–t path

for this instance. Now suppose we replace each edge cost c_e by its square, c_e^2, thereby creating a new instance of the problem with the same graph but with different costs. Prove or disprove the following claim: P must be a minimum-cost s–t path in this new instance.

4. [10 pts] Let S be an alphabet for which

$$\max_{x \in S} (f_x) < 2 \min_{x \in S} (f_x),$$

where f_x is the frequency of symbol x. Prove that if $|S| = 2^k$, then Huffman algorithm will give each symbol in S a codeword of length exactly k.

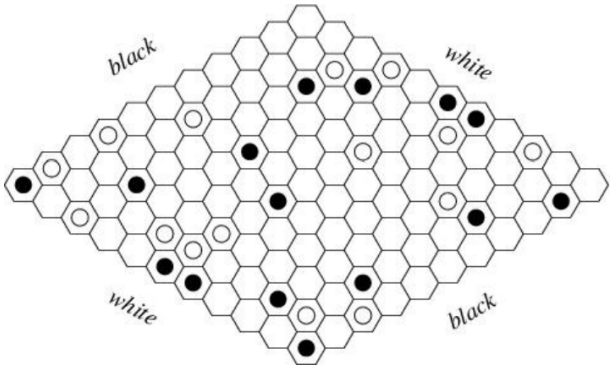

Figure 1: An illustration of a Hex game in progress, with $n = 11$.

5. [10 pts] The game of Hex is played on an $n \times n$ board composed of hexagonal cells. Two players take turns placing black and white stones on the cells (see Figure 1 for an illustration of a hex game in progress). If the upper left and lower right sides are connected to each other by a path of black stones, the black player wins. If the lower left and upper right sides are connected to each other by a path of white stones, the white player wins. Formalize the problem of detecting when one of the players has won and show how Union-Find can help with this problem. Provide some details as to the resulting running-time of your approach.

6. [10 pts] After observing the following number of occurrences of the following letters in a document: A: 2, B: 1, C: 3, D: ?, E: 5, F: 15, G: 15, I computed the frequencies of each of these letters by dividing the number of each of their occurrences by the total number of occurrences (which total = 41+?). Then I ran Huffman's algorithm, which produced the encoding: A: 01111, B: 01110, C: 0110, D: 00, E: 010, F: 10, G: 11.

a. Draw the Huffman tree for this code.

b. What are the possible (integer) values of the number of occurrences of the letter D? If you cannot give all the values, for partial credit, give at least one value that could produce this encoding. Justify your answer.