CSCE-629 Analysis of Algorithms

Fall 2017

Instructor: Dr. Jianer Chen
Office: HRBB 315C
Phone: 845-4259
Email: chen@cse.tamu.edu
Office Hours: T,Th 2:00pm-3:30pm

Teaching Assistant: Qin Huang
Office: HRBB 315A
Phone: 402-6216
Email: huangqin@email.tamu.edu
Office Hours: T,Th 9:00am-12:00noon

Assignment # 2
(Due October 10, 2017)

1. Design an algorithm for the following problem: given n line segments l_1, l_2, ..., l_n, either find two segments that intersect, or report that no two segments in the input intersect. You should make your algorithm as efficient as you can.

2. Write the psuedo-code for the Dijstra's algorithm that solves the SINGLE-SOURCE SHORTEST PATH problem. Analyze the complexity of the algorithm (you can assume that the algorithm uses a heap for fringes and you can use your results in Homework #1 directly). Give a formal proof that the algorithm works correctly when the edge weights are all non-negative.

3. This question is to convince you that Dijstra's algorithm may not work correctly if negative edges are allowed in a graph. Construct an instance for the SINGLE-SOURCE SHORTEST PATH problem in which the graph has negative edges, and show that Dijstra's algorithm produces an incorrect solution for the instance.

4. Develop a linear-time (i.e., $O(n + m)$-time) algorithm that solves the SINGLE-SOURCE SHORTEST PATH problem for graphs whose edge weights are positive integers bounded by 10. (**Hint.** You can either modify Dijstra's algorithm or consider using Breath-First-Search.)

5. Consider an extended version of the SHORTEST-PATH problem. Suppose that you want to traverse from city s to city t. In addition, for some reason, you also need to pass through cities x, y, and z (in any order) during your trip. Your objective is to minimize the cost of the trip. The problem can be formulated as a graph problem as follows: Given a positively weighted directed graph G and five vertices s, t, x, y, z, find a path from s to t that contains the vertices x, y, z such that the path is the shortest over all paths from s to t that contain x, y, z, assuming that these paths are allowed to contain repeated vertices and edges. Develop an $O(m \log n)$-time algorithm for this problem.