University of Washington October 10, 2024
Department of Computer Science and Engineering
CSE 417, Autumn 2024

Homework 3, Due Friday, October 18, 11:59 PM, 2024

Turn in instructions: Electronics submission on GradeScope. Submit as a PDF, with each problem on a separate page. Algorithms should be described in pseudo-code with some of the steps in English. You can use algorithms described in class such as Breadth First Search and Depth First Search at a high level, e.g., "Use Breadth First Search to . . ." or "Compute a Depth First Search Tree" without giving the code for BFS or DFS.

Problem 1 (10 points):

Suppose that an n-vertex undirected graph $G = (V, E)$ has vertices s and t with the distance from s to t strictly greater than $n/2$. Show that there must exist some vertex v, not equal to either s or t, such that deleting v destroys all $s - t$ paths. (This could be phrased as: show that a graph with *diameter* strictly greater than $n/2$ has an *articulation point.*) Give an algorithm that finds v in $O(n + m)$ time, you can assume that you are given the vertices s and t that have separation of greater than $n/2$.

Problem 2 (10 points):

Consider a directed graph on n vertices, where each vertex has exactly one outgoing edge. This graph consists of a collection of cycles as well as additional vertices that have paths to the cycles, which we call the branches. Describe a linear time algorithm that identifies all of the cycles and computes the length of each cycle. You can assume that the input is given as an array A, where $A[i]$ is the neighbor of i, so that the graph has the edge $\langle i, A[i]\rangle$.

For clarity, make sure that you describe in English the main steps of your algorithm, and don't just provide code. Justify the correctness of your algorithm.

Problem 3 (10 points):

Let $P = \{x_1, \ldots x_n\}$ be points on the X-axis in increasing order, and R be a non-negative integer. Give an $O(n)$ time algorithm to determine the minimum number of intervals of length R to cover the points. Explain why your algorithm is correct. (This problem relates to Chapter 4 material on greedy algorithms, but should be doable before the material has been presented in class.)

Programming Problem 4 (10 points):

The purpose of this problem is to construct a random graph generator for use in other programming problems and to demonstrate an implementation of graphs using adjacency lists.. A random graph generator, given an input parameter n, picks "at random" a graph with n vertices. There are multiple different models of random graphs. We will consider the *edge density* model, where a parameter p gives the probability of each edge being present. This model is referred to as $\mathcal{G}_p^n$.

The undirected random graph generation problem is: given an integer n and a real number p with $0 \leq p \leq 1$, construct an undirected graph on n vertices where each edge $\{u, v\}$ has probability p of being in the set of edges E, and the probability of each edge is independent. Write a generator for $\mathcal{G}_p^n$ which given inputs n and p constructs a random undirected graph in adjacency list representation.

For this problem, write the graph generator and a routine to print the edges and vertices of a graph. Print the results a creating two different random graphs using $n = 10$ and $p = 0.2$. Submit your code as a PDF.

Programming Problem 5 (10 points):

Implement an algorithm for computing the diameter of an undirected graphs. The diameter is defined to be the maximum distance between a pair of vertices if the graph is connected, and infinite if the graph is not connected. We define the *finite diameter* to be the maximum distance between a pair of vertices that are in the same connected component. Your algorithms should run in $O(mn)$ time for $G = (V, E)$ with $|V| = n$, and $|E| = m$. The algorithm should take as input an undirected graph, and compute the diameter and finite diameter.

Using the random graph generator from problem 4, experiment with the diameter and finite diameter of random graphs. For a fixed n, look at what happens as you vary the value of p. There are a couple of different experiments to run, one on dense graphs, the other on sparse graphs.

For dense graphs, what are the values of p that lead to small diameter graphs. Diameter 1 only occurs with $p = 1$, since you need all the edges. For a fixed value of n, what is the smallest values of p where you first get diameter 2 and diameter 3. You should be able to use a value of n of about 1000 for this (but don't choose a value of n where this takes a long time to generate graphs or compute the diameter). Instructor update: My testing with $n = 1000$, $p = 0.1$ yielded a diameter of 3.

For sparse graphs, what are the values of p where the diameter becomes finite (i.e., when the graph becomes connected). What is the value of p that maximizes the finite diameter. You should use a moderately large value of n, probably about $n = 10,000$ (again, do not choose an n so large that it takes a long time to run the algorithm.) The "interesting" values of p will be small, for example, with $n = 10,000$ the range of interest for p is $0.0002 \leq p \leq 0.002$. Instructor update: I ran a series a series of runs for $n = 10,000$. Each of them about one minute each. For $p = .001$, I had one connected component and diameter 7, for $p = 0.0005$, 50 components and diameter 11, and for $p = .00025$, 909 components and diameter 24.

For this problem, provides graphs on how the quantities vary with different values of p for fixed values of n. Provide a brief discussion of your results. Submit your code as a PDF.