Homework 04

CS 624, 2022 Fall

Review the course homework policies before you start!

1. Problem 17.2-3 (p459)

   > Suppose we wish not only to increment a counter but also to reset it to zero (i.e., make all bits in it 0). Counting the time to examine or modify a bit as $\Theta(1)$, show how to implement a counter as an array of bits so that any sequence of n INCREMENT and RESET operations takes time $O(n)$ on an initially zero counter. (Hint: Keep a pointer to the high-order 1.)

2. Problem 22.1-1 (p592).

   > Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?

3. Problem 22.2-3 (p602) — Note, corrected in 3rd printing.

   > Show that using a single bit to store each vertex color suffices by arguing that the BFS procedure would produce the same result if **line 18** were removed.

4. Problem 22.2-4 (p602)

   > What is the running time of BFS if we represent its input graph by an adjacency matrix and modify the algorithm to handle this form of input?

5. Problem 22.2-9 (p602)

   > Let $G = (V, E)$ be a connected, undirected graph. Give an $O(V + E)$-time algorithm to compute a path in G that traverses each edge in E exactly once in each direction. Describe how you can find your way out of a maze if you are given a large supply of pennies.

6. Problem 22.3-8 (p611)

   > Give a counterexample to the conjecture that if a directed graph G contains a path from u to v and if $u.d < v.d$ in in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.

7. Problem 22.3-9 (p612)

   > Give a counterexample to the conjecture that if a directed graph G contains a path from u to v then any depth-first search must result in $v.d \leq u.f$.

8. Problem 22.5-1 (p620)

   > How can the number of strongly connected components of a graph change if a new edge is added?

9. Problem 22.5-3 (p620)

   > Professor Bacon claims that the algorithm for strongly connected components would be simpler if it used the original (instead of the transpose) graph in the second depth-first search and scanned the vertices in order of increasing finishing times. Does this simpler algorithm always produce correct results?