

**Practice problems (don't turn in):**

1. Dijkstra's algorithm: This is not covered in the lectures because it's the sort of thing many of you have seen before, possibly multiple times (GT undergrads see it at least 3 times in their algorithms, data structures, and combinatorics class). If you haven't seen it, or need a refresher, look at Chapter 4.4 of [DPV].
    - (a) What is the input and output for Dijkstra's algorithm?
    - (b) What is the running time of Dijkstra's algorithm using min-heap (aka priority queue) data structure? (Don't worry about the Fibonacci heap implementation or running time using it.)
    - (c) What is the main idea for Dijkstra's algorithm?
  2. [DPV] Problem 3.3 (Topological ordering example)
  3. [DPV] Problem 3.4 (SCC algorithm example)
  4. [DPV] Problem 3.5 (Reverse of graph)
  5. [DPV] Problem 3.8 (Pouring water)
-

**Problem 1 [DPV] Problem 3.15 (Computopia)**

Note, linear time means  $O(n + m)$  where  $n = |V|$  and  $m = |E|$ .

**Part (a):**

**Part (b):**

**Problem 2 [DPV] Problem 4.14 (shortest paths through  $v_0$ )**

A faster algorithm is worth more. Be sure to state/explain the running time of your algorithm. Hint: Use Dijkstra's algorithm as a black-box. How many runs of Dijkstra's algorithm do you need?

**Answer: (Explain your algorithm in words and analyze its running time. No pseudocode.)**

---

### Problem 3 Global Destination

In this problem: use the algorithms from class, such as DFS, BFS, Dijkstra's, connected components, etc., as a black-box subroutine for your algorithm. If you attempt to modify one of these algorithms you will not receive full credit, even if it is correct. Make sure to explain your algorithm in words. (Note, this is problem 3.22 in [DPV].)

Let  $G = (V, E)$  be a **directed** graph given in its adjacency list representation. A vertex  $v$  is called a **global destination** if every other vertex has a path to  $v$ .

**Part (a).** Give an algorithm that takes as input a directed graph  $G = (V, E)$  and a specific vertex  $s$ , and determines if  $s$  is a global destination. Your algorithm should have linear running time, i.e.,  $O(n + m) = O(|V| + |E|)$ .

**Part (b). More Global Destination:**

Given an input graph  $G = (V, E)$  determine if  $G$  has a global destination or not. The running time of your algorithm should still be  $O(|V| + |E|)$ . In this problem you are no longer given  $s$ , and you need to determine whether or not  $G$  contains a global destination.

---