Data Structures and Algorithms II
Fall 2018
Midterm

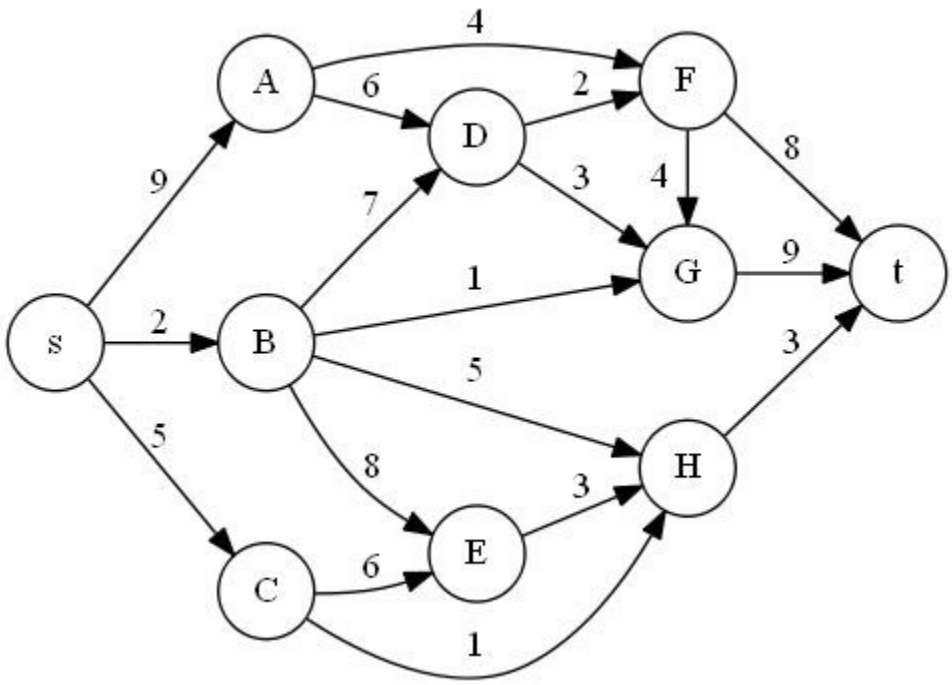
Name: _____

E-mail: _____

Write all of your answers directly within this test booklet. Use the backs of pages if necessary. This is an open-book, open-notes test. You should not use any electronic devices. You may use whatever scrap paper you like, but do not hand it in.

Please also write your name at the top of each page.

(1) Consider the following network flow graph (a.k.a. flow network):



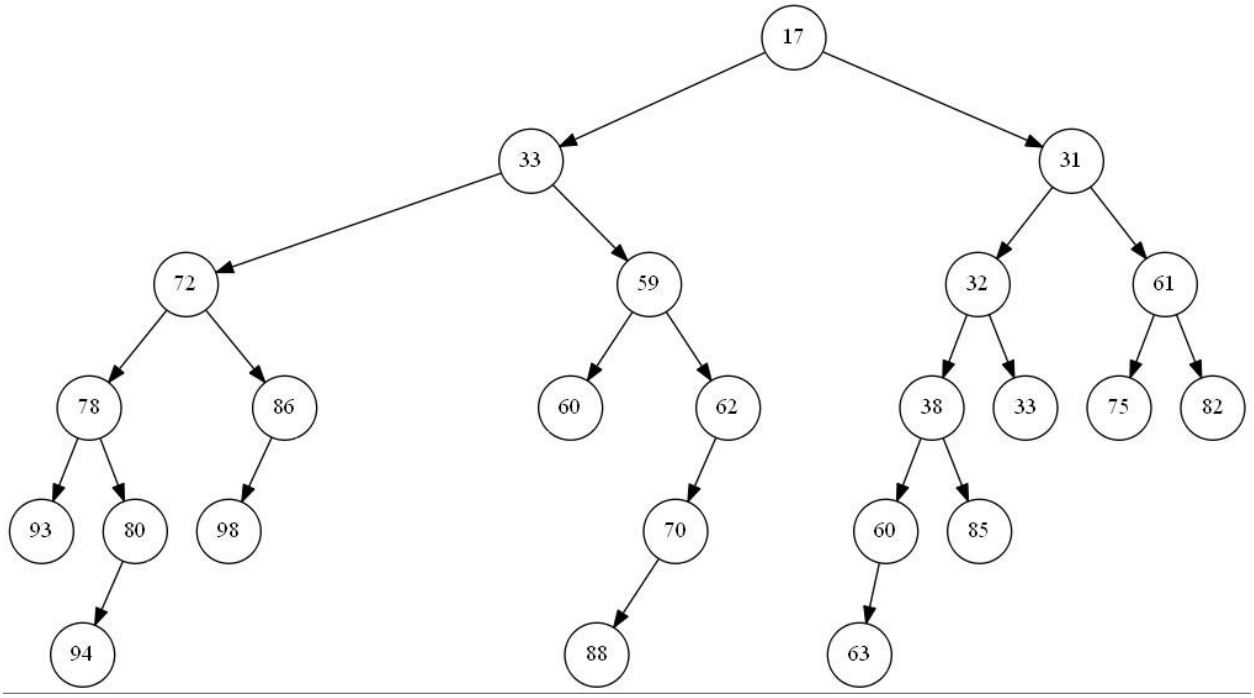
(a) You want to solve the maximum-flow problem using the Ford-Fulkerson method. You decide to implement the Ford-Fulkerson algorithm (i.e., for each iteration, you choose the augmenting path that leads to the biggest increase in flow). Specify all possible paths that you could choose as the first augmenting path (if there are ties, you have more than one choice). How much you could pump through each choice?

(b) Now suppose you decide to use the Edmonds-Karp algorithm instead (i.e., for each iteration, you choose the augmenting path that has the fewest number of edges). Specify all possible paths that you could choose as the first augmenting path. How much you could pump through each choice?

(c) Pump as much flow as you can through the flow network. If there are multiple possible ways that tie, choose any one of them. Draw below the final flow graph and residual graph that results after you achieve the maximum flow. Also state the maximum flow achieved.

(d) Use the max-flow, min-cut theorem to prove that you cannot beat the result you achieved in part (c). In other words, I want you to indicate an appropriate cut by drawing a line over the original graph. (You should have already stated the maximum flow in part (c), so you do not need to indicate anything in the space below.)

(2) Consider the following leftist heap:



(a) Label each node in the specified leftist heap above with its null path length. Use the conventions discussed in class (which match those of our textbook). You do not need to indicate the -1 values for non-existent children.

(b) Show the leftist heap that results when a deleteMin is applied to the specified leftist heap.

(c) Label each node in the leftist heap from part (b) with its null path length. Use the conventions discussed in class (which match those of our textbook). You do not need to indicate the -1 values for non-existent children.

(3) In class, we have examined the following pseudo-code for Dijkstra's algorithm:

```
WeightedPathLengthDijkstra ( Graph G, Vertex s )
     for each vertex v in G
          d_v ← ∞
          known_v ← FALSE
     d_s ← 0
     p_s ← NULL
     while there are still unknown vertices
          v ← the unknown vertex with the smallest distance
          known_v ← TRUE
          for each outgoing edge from v to vertex w
               if d_v + c_v,w < d_w
                    d_w ← d_v + c_v,w
                    p_w ← v
```

Assume a binary heap is used to implement the pseudo-code efficiently. Then consider applying Dijkstra's algorithm to a graph, G, containing a set of vertices, V, and a set of edges, E, all having positive weights. Briefly answer the following questions related to this scenario.

(a) What operation is used to find "the unknown vertex with the smallest distance"? What is the worst-case time complexity of this operation?

(b) What operation is used to apply the update "$d_w \leftarrow d_v + c_{v,w}$"? What is the worst-case time complexity of this operation?

(c) Consider the "if" statement "if $d_v + c_{v,w} < d_w$". If the comparison operator is changed from "<" to "<=" (less than or equal to), would the algorithm still be guaranteed to produce the optimal result? Explain your answer.

(d) Assume an extra parameter, "Vertex x", is added to the routine, and x represents a specific destination. Modify the Dijkstra pseudo-code so that the function stops after the optimal cost and path to x have been found. (Just write the necessary additional code next to the pseudo code above, with an arrow to show where the code needs to be inserted.)

(4) Answer the following questions about binary heaps, leftist heaps, and binomial queues (each supporting deleteMin, as discussed in class).

(a) Insert the numbers 1, 2, 3, 4, and 5, in that order, into an initially empty binary heap. Show the state of the binary heap after each insertion.

(b) Insert the numbers 1, 2, 3, 4, and 5, in that order, into an initially empty leftist heap. Show the state of the leftist heap after each insertion.

(c) Insert the numbers 1, 2, 3, 4, and 5, in that order, into an initially empty binomial queue. Show the state of the binomial queue after each insertion.

(5) Briefly answer the following questions about graph algorithms.

(a) Suppose you create a depth-first-search spanning tree for an undirected graph, starting with a vertex that turns out to be an articulation point. Is it possible that the root of the tree will have only one child? Explain your answer.

(b) Explain how the notions of cuts and crossing edges are relevant to proving the optimality of Prim's algorithm for determining a minimum spanning tree of a weighted, undirected graph.

(c) Can a topological sort be applied to disconnected directed acyclic graphs? (By disconnected, I mean not connected, i.e., there are separate islands of nodes in the graph that are not connected to each other.) Explain your answer.

(d) Is it necessarily the case that every instance of the maximum flow problem for a flow network can be reduced (i.e., mapped) to an instance of the maximum bipartite matching problem? Explain your answer.

(e) If the Bellman-Ford algorithm for solving the single-source, shortest-path problem is applied to a graph that is guaranteed to have no negative cost edges, but may have cycles, is it guaranteed to find the optimal paths to all nodes? Explain your answer.