CS 1501: Algorithm Implementation
Fall 2019

Practice Final Examination

November 2019 Suggested: 60 minutes

Name: _____ **Username (abc123):** _____

Instructions:

- This is a closed-book, closed-notes practice exam. It is recommended that you take it without using any outside resources, print or electronic.

Problem 1: Carefully read each prompt and select the best answer for each question.

i. Which PQ operation requires the data structure to be indexable?

(a) `update`
(b) `removeMax`
(c) `insert`
(d) `getMax`

ii. Which algorithm can compute the MST of a graph using a standard heap and no other secondary data structures?

(a) Dijkstra's
(b) Kruskal's
(c) Eager Prim's
(d) Lazy Prim's

iii. Which is faster in practice, Karatsuba's or the grade school multiplication algorithm?

(a) Each can be faster, depending on the input
(b) Grade school
(c) Karatsuba's
(d) None of the above

iv. In a 2048-bit RSA public key (n, e), what is the bitsize of n?

(a) 4096 bits
(b) 512 bits
(c) 1024 bits
(d) 2048 bits

v. Which of the following is the equality used by Euclid's algorithm, assuming $a > b$?

(a) $\gcd(a, b) = \gcd(b, a \bmod b)$
(b) $\gcd(a, b) = \gcd(a, b \bmod a)$
(c) $\gcd(a, b) = \gcd(a, a \bmod b)$
(d) $\gcd(a, b) = \gcd(b, b \bmod a)$

Problem 2: Consider a graph with 9 vertices (0-8). Give the contents of the ID and size arrays of a Union Find data structure after performing the following union operations. Use a weighted tree approach to implementing Union Find. You do not need to consider path compression. Resolve ties by keeping the numerically lesser connected component ID.

```
union(0,3); union(1,5); union(5,7); union(8,6);
      union(2,6); union(3,4); union(8,3);
```

	0	1	2	3	4	5	6	7	8
ID:									
Size:									

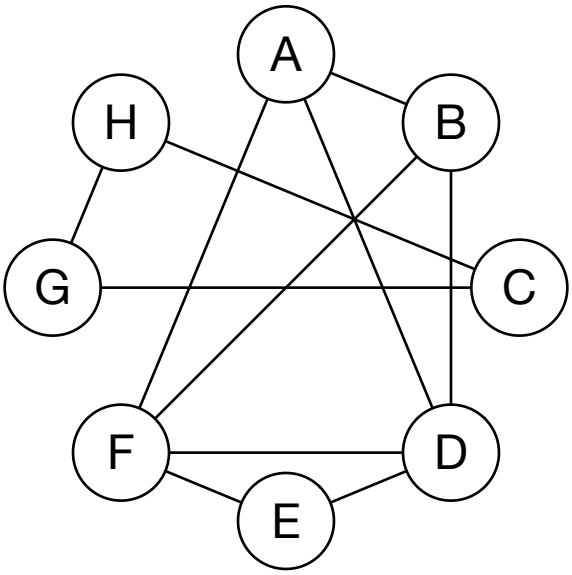
Problem 3: Which subproblems (multiplications) are completed when using Karatsuba's algorithm to multiply the following two binary values?

```
10110110 x 11001000
```

Problem 4: The following code is used in the book's implementation of KMP. Fill in the blanks so that the code works properly.

```
public KMP(String pat) {
    this.R = 256;
    this.pat = pat;

    // build DFA from pattern
    int m = pat.length();
    dfa = new int[R][m];
    dfa[pat.charAt(0)][0] = 1;
    for (int x = 0, j = 1; j < m; j++) {
        for (int c = 0; c < R; c++) {
            // Copy mismatch cases




            ____________________________________________________________;

        }
        // Set match case.




        ____________________________________________________________;

        // Update restart state.




        ____________________________________________________________;
    }
}
```

Problem 5: Use BFS to determine whether the following graph is connected. Start from vertex A, and assume neighbors are seen in alphabetical order.



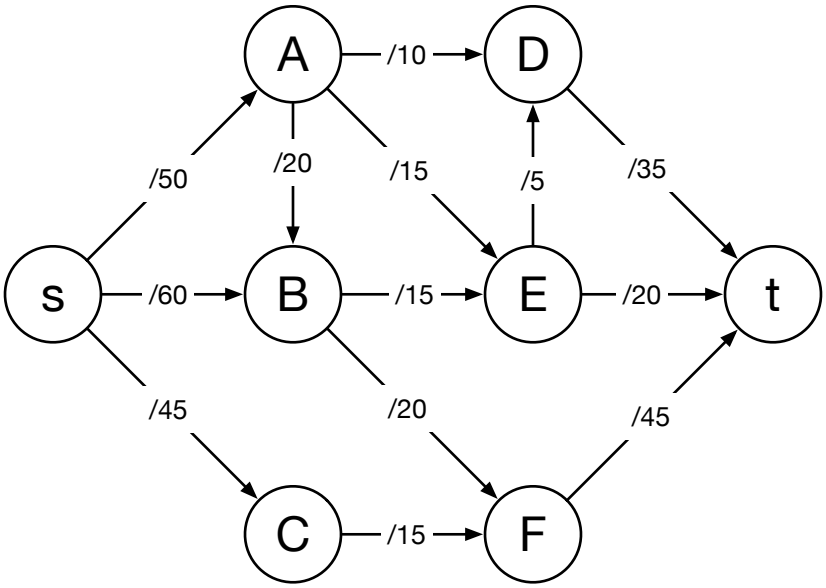
i. List the vertices that are visited and the order in which they are visited.

ii. Is the graph connected?

iii. What is the runtime of this BFS algorithm if the graph is represented as an adjacency list?

iv. What is the runtime of this BFS algorithm if the graph is represented as an adjacency matrix?

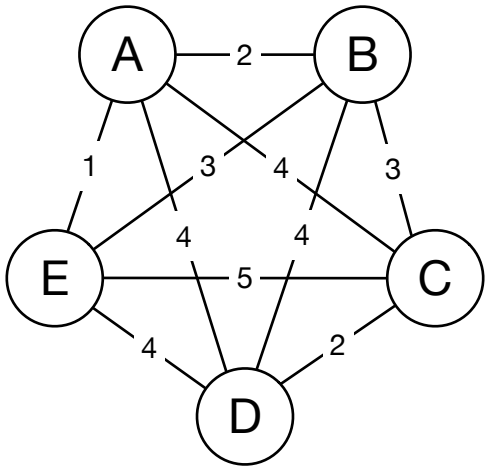
Problem 6: Use Edmonds-Karp to find the min s, t-cut of the following graph. Assume neighbors are seen in alphabetical order.



i. List the augmenting paths that you find and the order in which you find them. For each, state by how much flow is increased.

ii. State the size of the min cut and give a set of edges that comprises a min cut.

Problem 7: Recall the nearest-neighbor and MST approximation heuristics for the Traveling Salesman Problem. Compute a TSP tour using each of these heuristics over the following graph of cities.



i. Use the nearest-neighbor heuristic, starting from A, to find a tour. Resolve ties alphabetically (e.g., choose A instead of C if they are otherwise tied). Give the order in which the cities are visited and the total length of the tour.

ii. Use the MST heuristic to find a tour. Find an MST using Prim's algorithm starting from A, again resolving ties alphabetically. Use DFS starting from A to determine the shortcut tour. Give the order in which the cities are visited and the total length of the tour.