# Spring 2018 COP 3503 Section 1 Exam #2, March 22, 2018

## Last Name: _____, First Name: _____

1) (15 pts) Huffman Coding

Consider a file that contains the following eight distinct characters (each encoded using 3 bits) with the following frequencies:

| Character | Frequency | Huffman Code |
|---|---|---|
| A | 15000 | |
| B | 24000 | |
| C | 12000 | |
| D | 8000 | |
| E | 27000 | |
| F | 5000 | |
| G | 9000 | |
| H | 20000 | |

a) (5 pts) Draw a Huffman Coding Tree for this file below.

b) (5 pts) In the chart above, fill in the corresponding Huffman codes for each character in the file based on the tree you drew in part a. Please use '0' = left, '1' = right.

c) (5 pts) How many bits would be saved using the Huffman coding on this file, assuming that the code itself took 100 bits to store?

2) (15 pts) Network Flow Algorithm Design

Consider the problem of assigning teachers to classes at Triple Threat University. At the university, there are three sections of each class offered, each at the same exact time. All different classes are offered at different times. Each professor the university hires has a maximum number of courses they can teach and a list of courses for which they are proficient in teaching and are available during the time period when the class is taught. (Unlike some places, Triple Threat refuses to put a teacher in a classroom for which they aren't proficient.) Given all of this data, design an efficient algorithm to determine if it is possible or not for Triple Threat University to assign professors to each of its classes. After describing your algorithm, create a small example and draw the corresponding network flow graph for that one example.

3) (15 pts) Dijkstra's Algorithm Trace

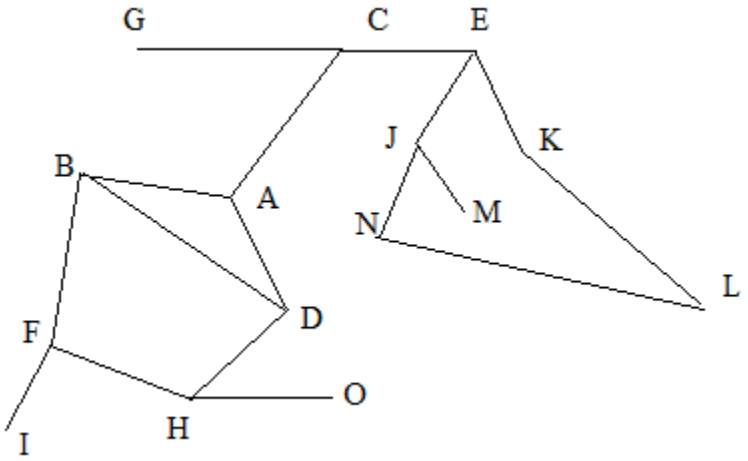
Trace through Dijkstra's algorithm on the graph described below **using C as the source vertex.** The last line of your chart should display the shortest distances from C to all of the vertices in the graph. In order to receive credit, you must properly fill in the chart, which shows incremental updates to the distance array. (Note: Each edge shown below is a directed edge.)

| | | | | |
|---|---|---|---|---|
| AB 4 | AD 12 | AE 10 | AF 5 | AG 3 |
| BA 5 | BD 15 | BE 20 | BG 30 | |
| CA 12 | CB 6 | CD 25 | CF 18 | |
| DA 5 | DB 9 | DE 6 | DF 12 | DG 15 |
| EA 20 | EB 18 | ED 1 | EG 6 | |
| FD 12 | FE 1 | | | |
| GD 4 | GE 5 | GF 1 | | |

| Add to S | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| C | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

4) (10 pts) Graphs – Depth First Search

Show the order in which the vertices get visited during a depth first search of the graph shown below, starting at vertex A. When determining the order in which to make the recursive calls from a given node, always go in alphabetical order.




___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___

5) (10 pts) Greedy Algorithms

The 0-1 Knapsack problem is as follows: you are given a list of items, each item has an integer weight and integer value. The goal of the problem is to choose a subset of the items which have a sum of weights less than or equal to a given W with a maximal sum of values. For example, if we had the following five items (each in the form (weight, value)): $I_1$(6, 13), $I_2$(4, 10), $I_3$(1, 1), $I_4$(8, 12), and $I_5$(5, 9) and W = 13, then our maximal achievable value is 13 + 10 + 1 = 24, corresponding to the subset containing the first three items, which weigh 11 units, which is less than or equal to our weight limit of 13 units. A proposed greedy algorithm to solve the problem is as follows: (1) sort the items by per unit value. Go through the list of items in this order, taking each item as long as there is room to do so. (For this example, when we sort by per unit value, the ordering would be $I_2$, $I_1$, $I_5$, $I_4$, and $I_3$. Then, we'd take $I_2$, followed by $I_1$, skip $I_5$ and $I_4$ because enough weight wasn't left, then take $I_3$.) This algorithm is incorrect. Create a single example (a set of items with weights and values) for which this algorithm fails. In your example, show what answer the algorithm will produce and show a subset that is more valuable (but also within the weight limit.)

6) (15 pts) Graphs – Breadth First Search

Consider the problem of attempting to get from the top left corner to the bottom right corner of a square grid in the minimum number of moves where each square is marked as '.' (valid) or 'X' (invalid). (Both the top left and bottom right corners are always guaranteed to be valid squares.) The valid moves are up, down, left and right and the top left corner is coordinate (0, 0) and the bottom right coordinate is (n-1, n-1), where n is given in the input. Complete the code below to correctly solve the problem. All of the I/O is handled and you just have to fill in portions of the method that conducts the breadth first search. (Note: The method is designed only to return the shortest distance to the bottom right corner, not to necessarily mark all of the shortest distances from the top left to all other squares.) The method returns -1 if there is no valid path.

```
import java.util.*;

public class bfs_maze {

    public static int n;
    public static char[][] grid;
    final public static int[] DX = {-1,0,0,1};
    final public static int[] DY = {0,-1,1,0};
```

```
    public static void main(String[] args) {
        Scanner stdin = new Scanner(System.in);
        int nC = stdin.nextInt();

        for (int loop=0; loop<nC; loop++) {
            n = stdin.nextInt(); // Board from (0,0) to (n-1,n-1).
            grid = new char[n][];
            for (int i=0; i<n; i++)
                  grid[i] = stdin.next().toCharArray();
            System.out.println(bfs(0, n*n-1));
        }
    }

    public static int bfs(int start, int end) {

        LinkedList<Integer> q = new LinkedList<Integer>();
        int[][] dist = new int[n][n];
        for (int i=0; i<n; i++)
            Arrays.fill(dist[i], -1);
        dist[start/n][start%n] = 0;
        q.offer(start);

        while (q.size() > 0) {

            int cur = q.poll();

            if (cur == _____ ) return dist[ ______ ][ ______ ];


            for (int i=0; i<DX.length; i++) {

                int x = _____________________;

                int y = _____________________;

                if (!inbounds(x,y) || dist[x][y] != ___ || grid[x][y] == ___)
                    continue;

                q.offer( _____________________ );

                dist[x][y] = dist[ ______ ][ ______ ] + 1;
            }
        }

        return -1;
    }

    public static boolean inbounds(int tryX, int tryY) {
        return tryX >= 0 && tryX < n && tryY >= 0 && tryY < n;
    }
}
```

7) (15 pts) Greedy Algorithm Design

A couple wants to buy $n$ major items. They will buy one item per month. They will buy all of their items at the CrazySuperStore. Due to the current economy, it is known that at some point in the future there will be a single price hike on all items at the same time, but due to the uncertainty of economic processes, it is not known when that price hike will occur. For each of the $n$ items the couple wants to buy, both the current price, $a_i$ and the future price, $b_i$ ($b_i > a_i$), are known. (Note that the individual price hikes on items may vary. For example, one item may go from \$100 to \$105 while another might go from \$300 to \$400. They only assumption you may make about the prices is that the second price is strictly higher than the first and that you have access to all before/after prices.) Devise a greedy strategy so that the couple is guaranteed to minimize the amount of money they spend on the items. Assume that no matter what, the couple will buy all items. Clearly describe your strategy and intuitively (or formally) prove why it's optimal.

8) (5 pts) What types of cakes are served at "Nothing Bundt Cakes"? _____

**Scratch Page – Please clearly mark any work on this page you would like graded.**