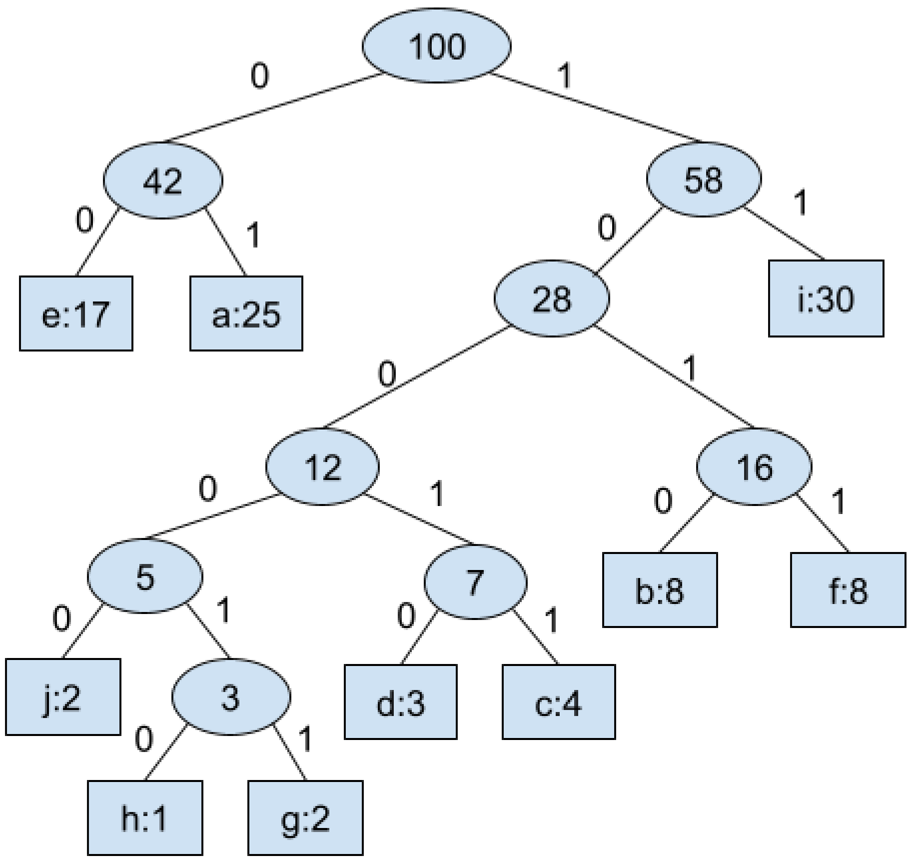
COSC581 - Algorithms
Spring 2023
Homework #5 Solutions

1. Construct a huffman tree for the following set of frequencies and show the optimal encoding:



Huffman encoding: {e:00, a: 01, i:11, b: 1010, f: 1011, j: 10000, d: 10010, c:10011, h: 100010, g: 100011}

**Note: flipping 0's and 1's produces the same code (also acceptable based on tree).*

2. Suppose we must make a payment of n cents using only pennies, dimes, and quarters. We want to find the smallest set of coins possible with the total value n.
    a. Prove that the greedy approach does not always work by finding a counterexample. Give an amount n, the set of coins that the greedy approach yields for this amount, and a smaller set of coins with the same total value.

<u>Counterexample</u>: Let n=40 cents, then the greedy algorithm would choose from the set {1, 10, 25} the number 25 because it is the largest number < 40. It would then choose the number 10 because 25+10 < 40, and finally it would choose 5 1's because 35+5 <= 40. This results in 1 quarter, 1 dime and 5 pennies (I am assuming no nickels based on how the problem is worded) for a total of 6 coins. The optimal solution however, would be 4 dimes for a total of 4 coins. So in this example, the greedy algorithm did not choose the most optimal solution.

    b. Is there a set of coin denominations such that a greedy algorithm always yields an optimal solution? If so, provide such a system with at least three denominations.

Yes there is a set of coin denominations where the greedy algorithm always produces an optimal solution. One such example is to include the nickel in with pennies, dimes and quarters (S = {1,5,10,25}). This will work with the greedy algorithm because the set is a matroid and matroids imply that the greedy algorithm will yield an optimal solution (Lemma 16.7 and Thm 16.11 in the textbook).

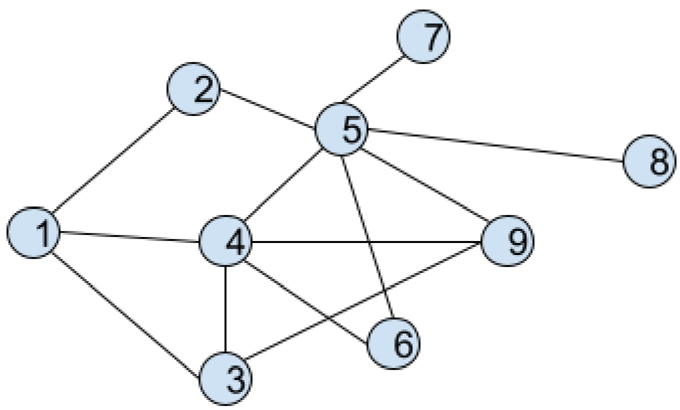
Proof:

Let S= {1,5,10,25}

Then,
1. S is a finite set (n=4)
2. Let x belong to S. Let A be a subset of S such that x belongs to A. Let B be a subset of S such that B is a subset of *I* and A is a subset of B. Then x belongs to A => x belongs to B => x belongs to *I*.
3. Since some combination of the previous x_{i-1} should always add to equal x_i, we can see that there exists y belonging to B \ A that will satisfy our weight function *I*, and thus A union {y} belongs to *I*.

Thus, since (S, *I*) is a matroid, we can say that the greedy algorithm will produce an optimal solution for the coin problem for S based on Thm 16.11. ▮

3. Describe the graph shown below using each of the four types of graph representations discussed in class (adjacency matrix, adjacency list, incidence matrix, and a simplified DIMACS format).The simplified DIMACS format is as follows: one header line that consists of a tab separated pair of integers representing the number of vertices and edges in the graph, and then a series of lines consisting of (u,v) pairs of tab separated vertex indices that represent an edge between vertices u and v.



Adjacency matrix:

Nodes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	1	1	0	0	0	0	0
2	1	0	0	0	1	0	0	0	0
3	1	0	0	1	0	0	0	0	1
4	1	0	1	0	1	1	0	0	1
5	0	1	0	1	0	1	1	1	1
6	0	0	0	1	1	0	0	0	0
7	0	0	0	0	1	0	0	0	0
8	0	0	0	0	1	0	0	0	0
9	0	0	1	1	1	0	0	0	0

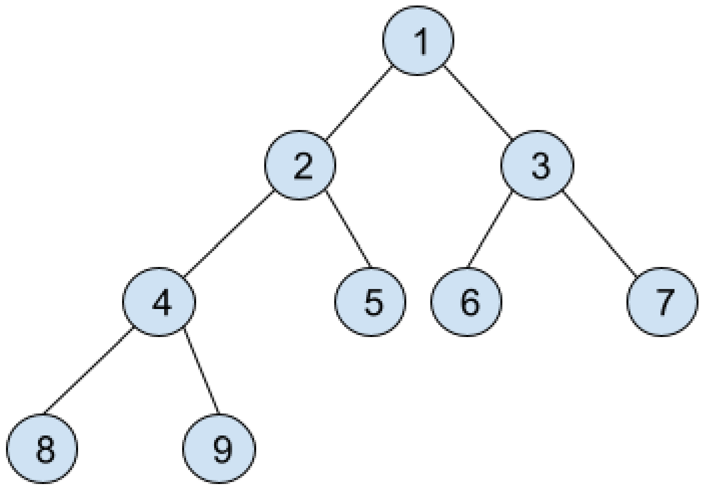
Adjacency List:
[[2,3,4], [1,5], [1,4,9], [1,3,5,6,9], [2,4,6,7,8,9], [4,5], [5], [5], [3,4,5]]

Incidence Matrix:

-	V1	V2	V3	V4	V5	V6	V7	V8	V9
E1 (1↔2)	1	1	0	0	0	0	0	0	0
E2 (1↔3)	1	0	1	0	0	0	0	0	0
E3 (1↔4)	1	0	0	1	0	0	0	0	0
E4 (2↔5)	0	1	0	0	1	0	0	0	0
E5 (3↔4)	0	0	1	1	0	0	0	0	0
E6 (3↔9)	0	0	1	0	0	0	0	0	1
E7 (4↔5)	0	0	0	1	1	0	0	0	0
E8 (4↔6)	0	0	0	1	0	1	0	0	0
E9 (4↔9)	0	0	0	1	0	0	0	0	1
E10 (5↔6)	0	0	0	0	1	1	0	0	0
E11 (5↔7)	0	0	0	0	1	0	1	0	0
E12 (5↔8)	0	0	0	0	1	0	0	1	0
E13 (5↔9)	0	0	0	0	1	0	0	0	1

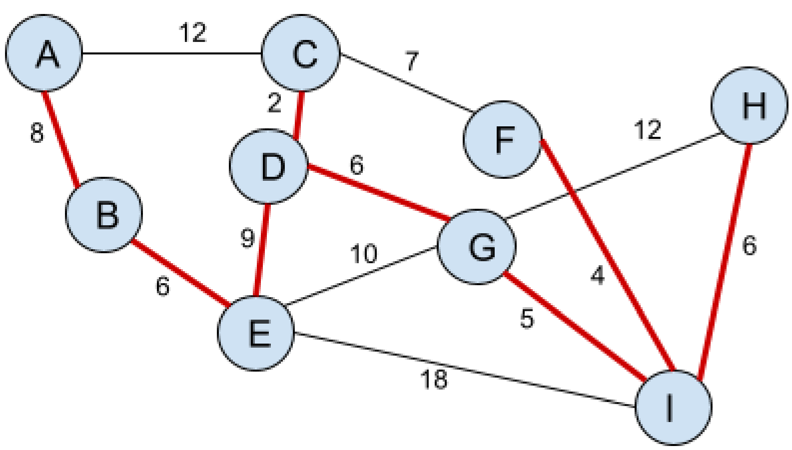
Dimacs Format:
9 13
1 2
1 3
1 4
2 5
3 4
3 9
4 5
4 6
4 9
5 6
5 7
5 8
5 9

4. Given the following graph, show the order in which the nodes will be traversed for both a DFS (depth first search) and BFS (breadth first search).



BFS: 1, 2, 3, 4, 5, 6, 7, 8, 9
DFS: 1, 2, 4, 8, 9, 5, 3, 6, 7

5. For the following graph, name one algorithm to construct the minimum spanning tree. List the order in which you would add edges to the tree for your chosen algorithm.



**Note: I renamed the nodes from 1-9 to A-I for the solutions*

Kruskals:

1. Create a forest F of trees:
    a. {CD, 2}, {FI, 4}, {GI, 5}, {BE, 6}, {DG, 6}, {HI, 6}, {CF, 7}, {AB, 8}, {DE, 9}, {EG, 10}, {AC, 12}, {GH, 12}, {EI, 18}
2. While edges do not form a cycle, add to MST:
    a. {CD, 2}, {FI, 4}, {GI, 5}, {BE, 6}, {DG, 6}, {HI, 6}, {AB, 8}, {DE, 9}

Prims:

1. Initialize the tree with a single vertex (chosen arbitrarily - I chose A).
2. While there are vertices not yet visited, choose lowest weight edge from current vertice and add to queue:
    a. Visit A: q = {B(8), C(12)}, MST= {AB}
    b. Visit B: q = {E(6), C(12)}, MST = {AB, BE}
    c. Visit E: q = {D(9), G(10), I(18), C(12)}, MST = {AB, BE, DE}
    d. Visit D: q = {C(2), G(6), E(9), G(10), I(18), C(12)}, MST = {AB, BE, DE, CD}
    e. Visit C: q = {F(7), A(12), G(6), E(9), G(10), I(18), C(12)}, MST = {AB, BE, DE, CD, CF}
    f. Visit F: q = {I(4), A(12), G(6), E(9), G(10), I(18), C(12)}, MST = {AB, BE, DE, CD, CF, FI}
    g. Visit I: q = {G(5), H(6), E(18), A(12), G(6), E(9), G(10), I(18), C(12)}, MST = {AB, BE, DE, CD, CF, FI, GI}
    h. Visit G: q = {D(6), E(10), H(12), H(6), E(18), A(12), G(6), E(9), G(10), I(18), C(12)}, MST = {AB, BE, DE, CD, ~~CF~~, FI, GI, DG}
    i. Visit D, E: visited
    j. Visit H: q = {H(6), E(18), A(12), G(6), E(9), G(10), I(18), C(12)}, MST = {AB, BE, DE, CD, ~~CF~~, FI, GI, DG, GH}
    k. Visit H: q = {I(6), H(6), E(18), A(12), G(6), E(9), G(10), I(18), C(12)}, MST = {AB, BE, DE, CD, ~~CF~~, FI, GI, DG, ~~GH~~, HI}
    l. Visit I, H, E, A…C: visited

Final MST = {AB(8), BE(6), DE(9), CD(2), FI(4), GI(5), DG(6), HI(6)}