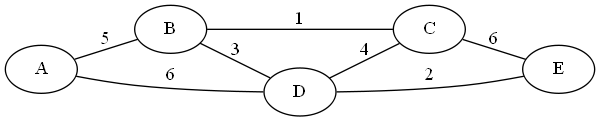
Name: \_\_\_Katie Prescott\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_4/29/16\_\_\_\_\_\_\_\_\_

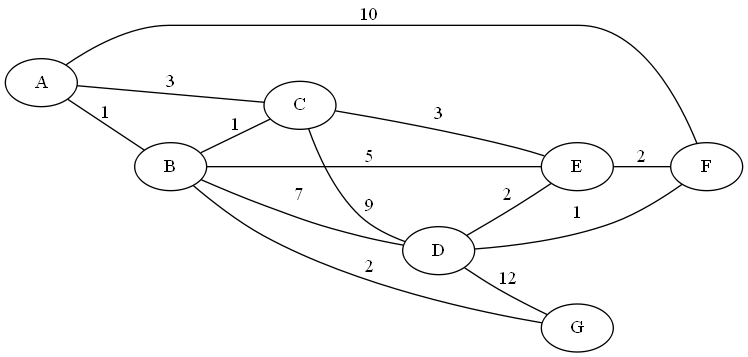
1. Apply Kruskal’s algorithm to find a minimum spanning tree of the graph below.

X x x

Bc de bd cd ab ad ce

1 2 3 4 5 6 6

**Include edges: AB, BC, BD, DE**

2. Use Dijkstra’s algorithm to find the shortest path between nodes **A** and **F**. (Start at A, end at F)

|  |  |
| --- | --- |
| **Tree Vertices** | **Remaining Vertices** |
| A(-, 0) | B(A, 1), C(A, 3), D(-, ∞), E(-, ∞), F(A, 10), G(-, ∞) |
| B(A, 1) | C(B, 1+1), D(B, 1+7), E(B, 1+5), F(A, 10), G(1+2) |
| C(B, 2) | D(B, 8), E(C, 2+3), F(A, 10), G(B, 3) |
| G(B, 3) | D(B, 8), E(C, 5), F(A, 10) |
| E(C, 5) | D(E, 5+2), F(E, 5+2) |
| D(E, 7) | F(E, 7) |
| F(E, 7) |  |
|  |  |
|  |  |

Path: **AB, BC, CE, EF**

**Distance: 7**

3.

a. Construct a Huffman tree for the following data:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Symbol** | A | B | C | D | \_ |
| **Frequency** | 0.4 | 0.1 | 0.2 | 0.15 | 0.15 |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| |**

[ B D ] \_ C A **ABD\_C**

[0.1 0.15] 0.15 0.2 0.4 **0/ \1**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ A .6**

**| | 0/ \1**

[ \_ C ] BD A  **.25 .35**

[.15 .2 ] .25 .4  **0 / \1 0 / \1**

**\_\_\_\_\_\_\_\_\_\_\_\_\_ B D \_ C**

**| |**

[ BD \_C ] A

[.25 .35 ] .4

b. Encode ABACABAD using the tree you generated for (a).

**0100011101000101**

c. Decode 100010111001010 using the tree you generated for (a).

**BAD\_ADA**

d. What compression gain (percent of improvement) do we get by using Huffman encoding instead of a fixed‐length encoding scheme (assume the fixed‐length scheme would require 3 bits for each character)?

.4 x 2 = .8 .8 + .2 + .4 + .3 + .3 = 2

.1 x 2 = .2 (3 - 2) / 3 x 100 = **33.3%**

.2 x 2 = .4

.15 x 2 = .3