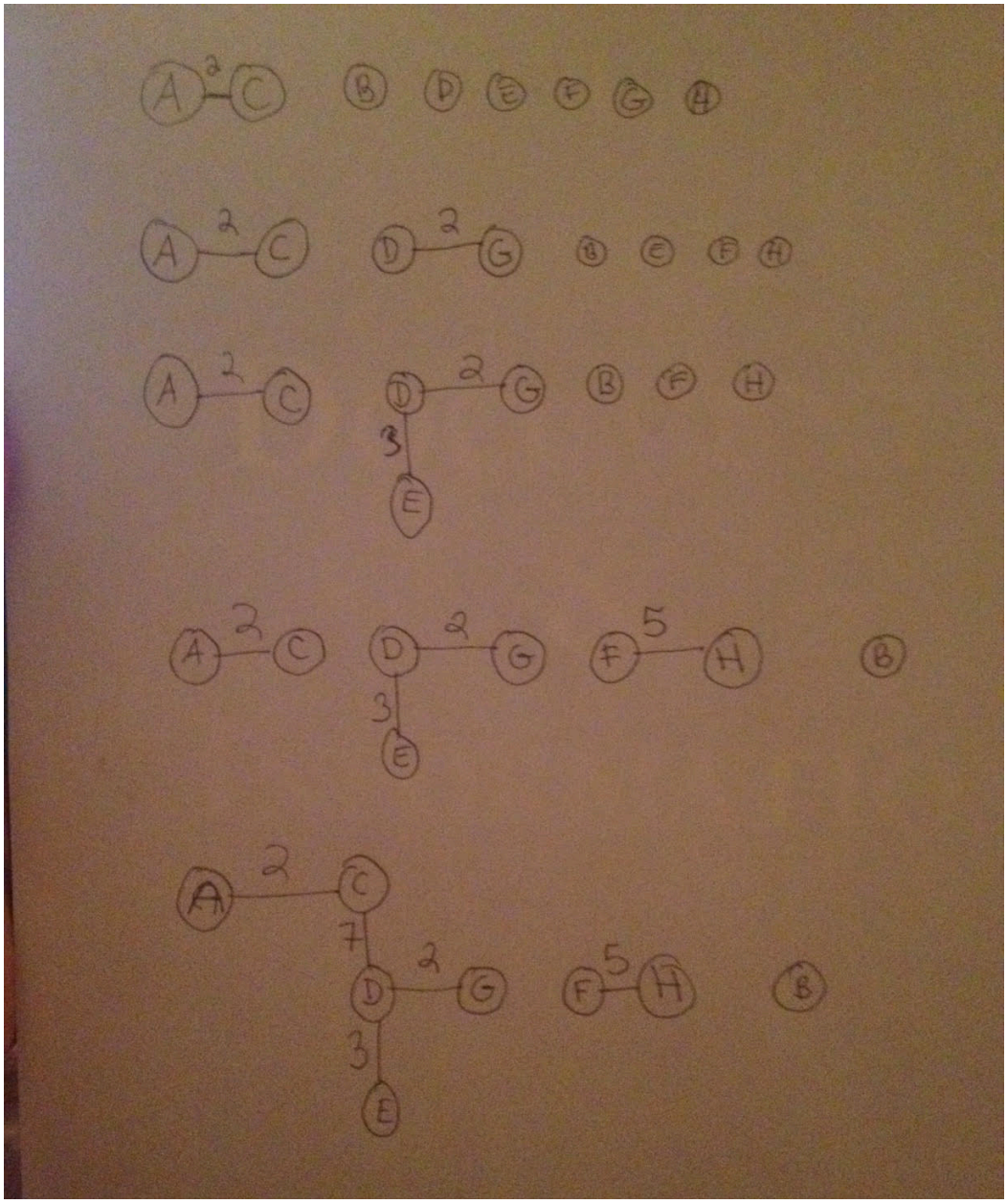
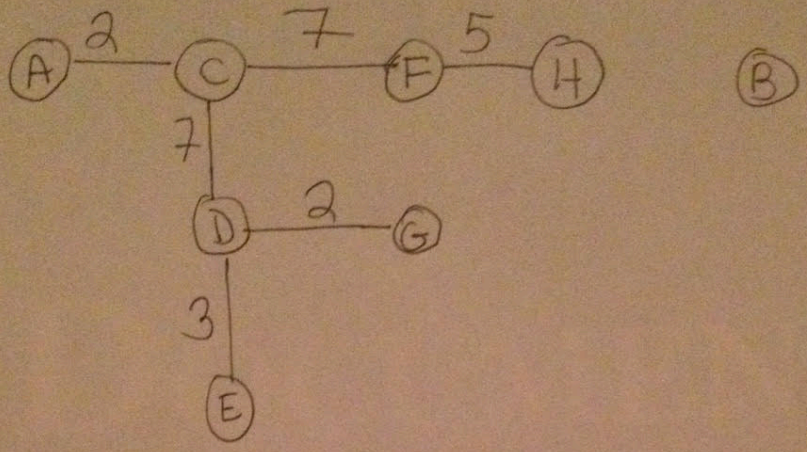
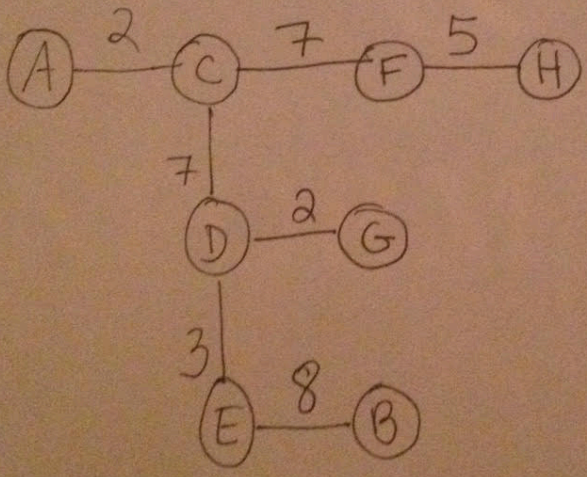
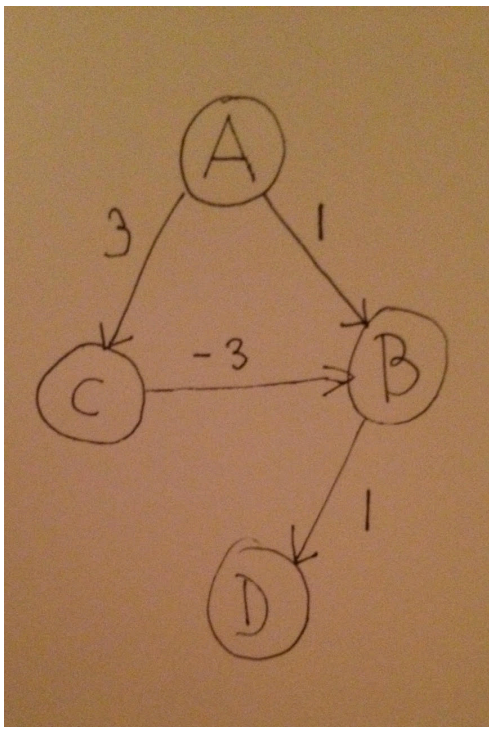
Problem 1


A 2 C
B
D
E
F
G
H
A 2 C
D 2 G
B
E
F
H
A 2 C
D 2 G
3
E
B
F
H
A 2 C
D 2 G
3
E
F 5 H
B
A 2 C
7
D 2 G
3
E
F 5 H
B


A
2
C
7
F
5
H
B
7
D
2
G
3
E


A
2
C
7
F
5
H
7
D
2
G
3
E
8
B

Problem 2



Distances	Queue
d(A) = 0, d(B) = inf, d(C) = inf, d(D) = inf	A
d(A) = 0, d(B) = 1, d(C) = 3, d(D) = inf	B, C
d(A) = 0, d(B) = 1, d(C) = 3, d(D) = 2	C, D
d(A) = 0, d(B) = 1, d(C) = 3, d(D) = 2	C
d(A) = 0, d(B) = 0, d(C) = 3, **d(D) = 2**	

At the end of this run, the distance to D is incorrectly marked as 2, instead of cost 1 by going A->C->B->D.

Problem 3

We will modify BFS for this. We can label every node as belonging either to group 0 or 1. The graph is bipartite if all neighboring pairs of nodes have opposite labels.

To do this, we basically do BFS but instead of computing distances, we assign labels. We give the start node a label of 0 and everything else a label of -1. When we examine the neighbors v of u, we check if they have the same label. If so, we return false. Otherwise, we assign v the opposite label (0 or 1) of u and add it to the queue. Since BFS runs of O(|V|+|E|) time, so does this modified version.