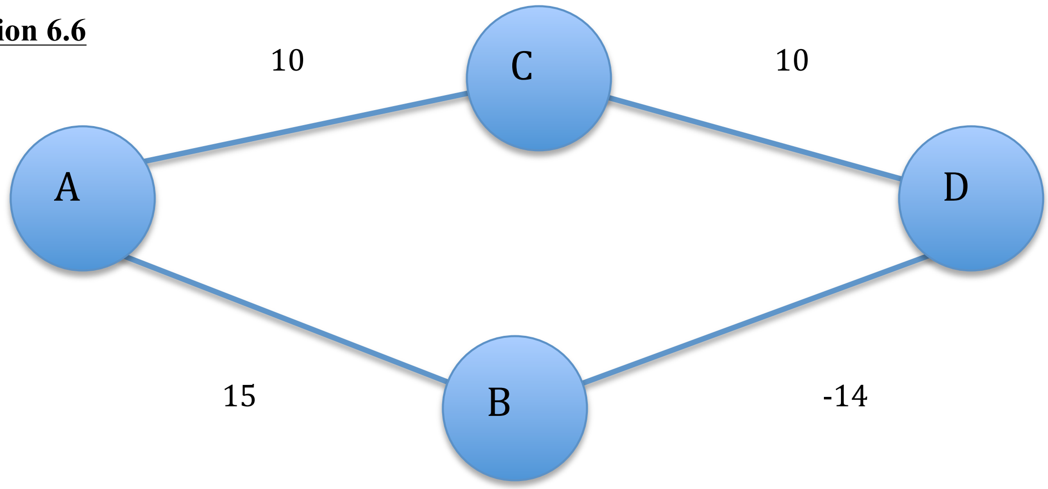
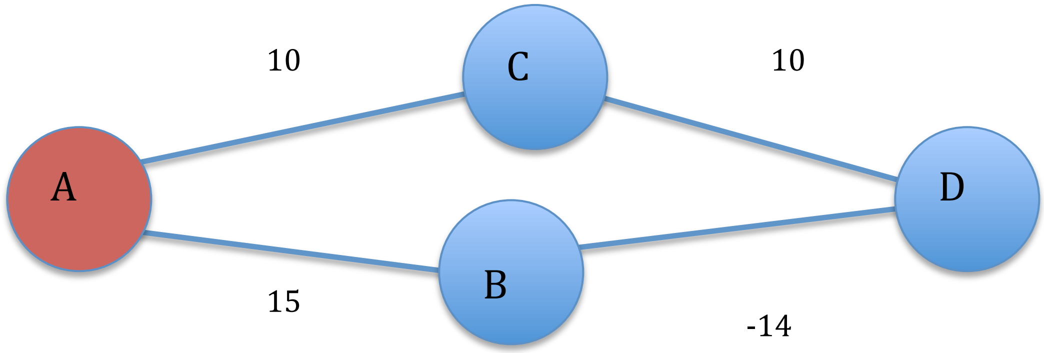
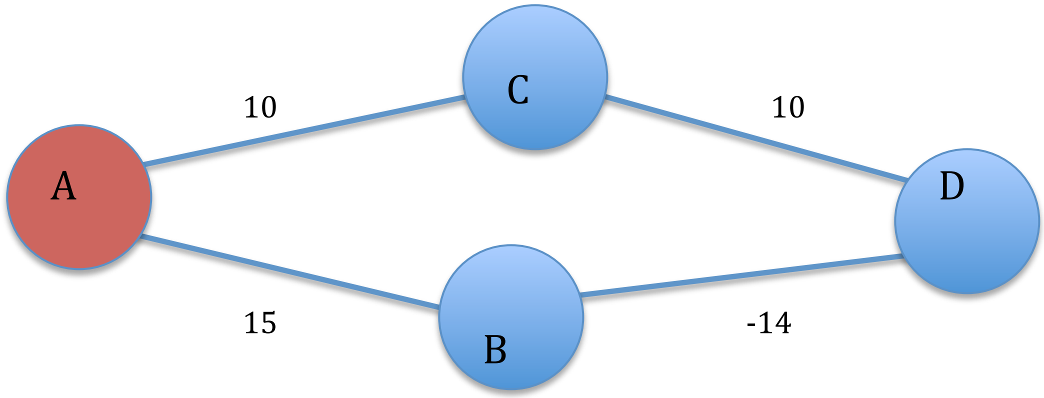
Adam Jablonski

CS260 – HW8

<u>Section 6.6</u>







10 + 10	15 + -14
A -> D = 20	A -> D =1

The Disjkstra algorithm states the shortest path is a weight of 20 however this is not true, the shortest path is 1. The Dijkstra algorithm can not handle negatives therefore the result is incorrect.

Section 7.1

```
Procedure Insert( int j, int j)
    # pull up list from map
    loc1 = First(map[i])
    loc2 = First(map[j])
    while loc1 != null
        loc1.Next()
        loc1.Next = node(j)

    while loc2 != null
        loc2.Next()
        loc2.Next = node(i)
    Map[i] = loc1
    Map[j] = loc2
```

```
Procedure Delete( int I, int j)
    loc1 = First(map[i])
    loc2 = First(map[j])

    while loc1.Next != null
        if loc1.value == j
            tmp = loc1
            tmp2 = loc1.Next()
            tmp.Next = node(j, tmp2)

    while loc2.Next != null
        if loc2.value = i
            tmp = loc2
            tmp = loc.Next()
            tmp.Next = node(j, tmp)

    Map[i] = loc1
    Map[j] = loc2
```

Section 7.2

Use a Matrix Representation:

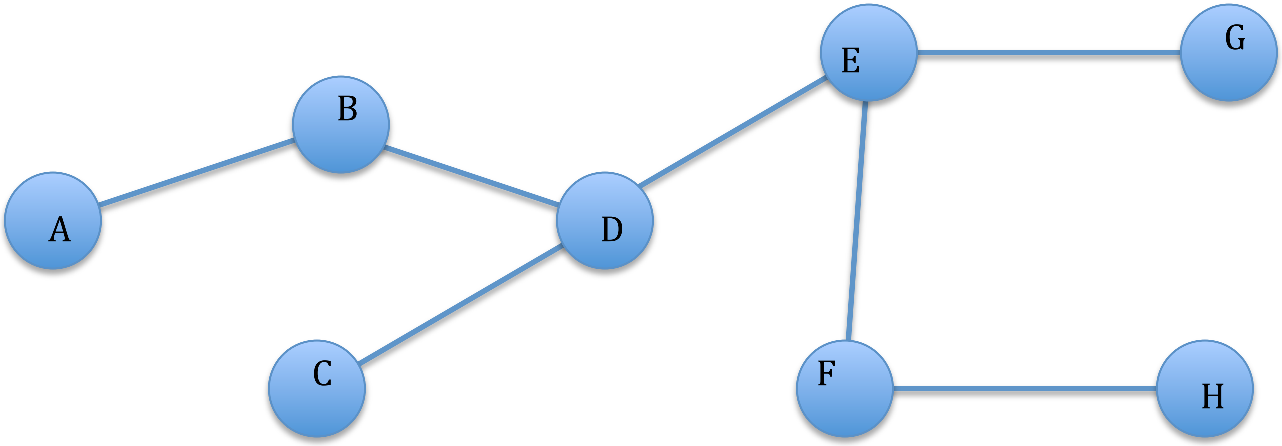
$$\begin{array}{ccccc} & A & B & C & D \\ A & 0 & 1 & 0 & 1 \\ B & 1 & 0 & 1 & 0 \\ C & 0 & 1 & 0 & 0 \\ D & 0 & 0 & 0 & 1 \end{array}$$

```
Procedure to DeleteFirstEdge(I, j)
    List[i][j] = 0
    List[j][i] = 0
```

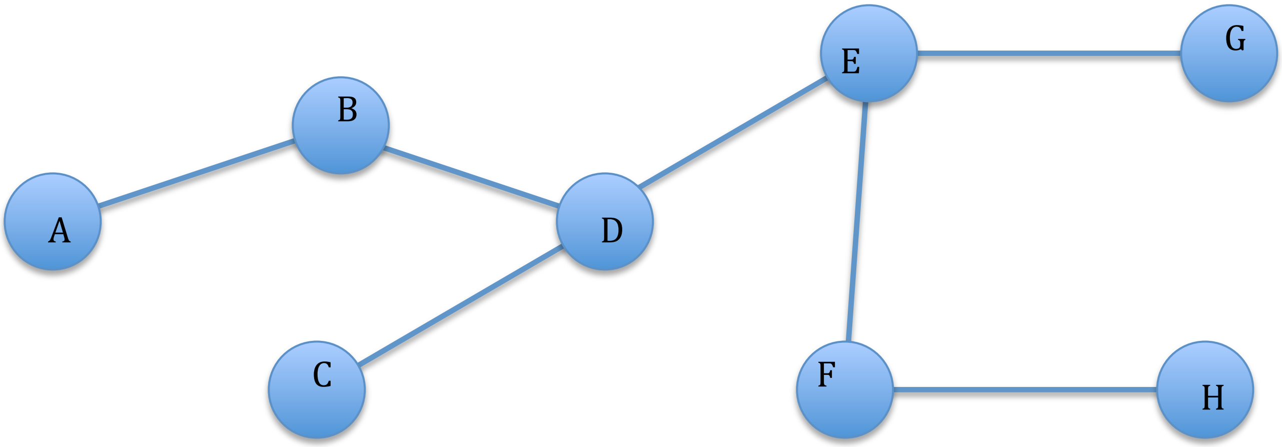
It is a constant time deletion because you always know where it is Alternativly you could just have a tree representation and have a data location to the first node and the second node. From there you would just change the first node to the second node and since you knew where they were already it would also be constant time.

Section 7.3

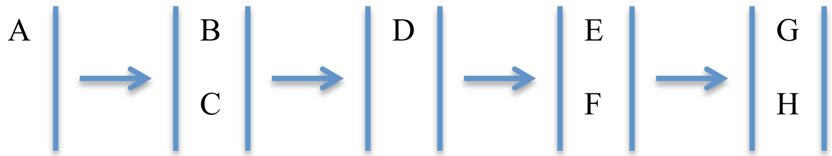
a) Weight 11



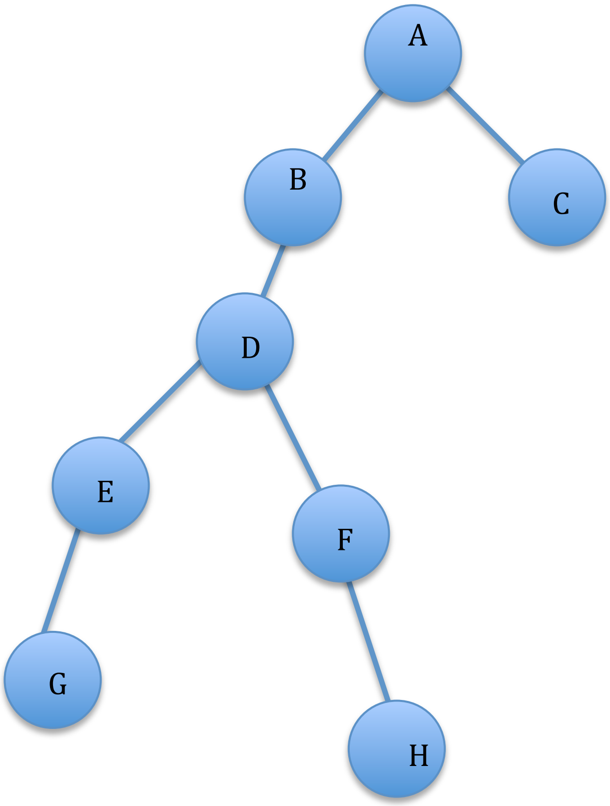
B)



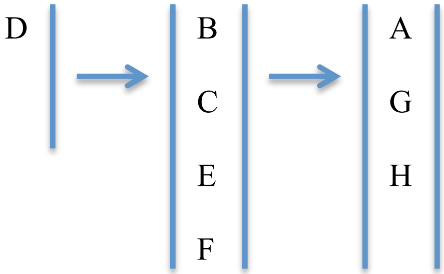
D_a)



Order: A B C D E F G H



D_b)



Order: D B C E F A G H