b) Although both algorithms produce the same result, one could close the edge between F and I instead of B and F. The total cost is still minimum.

Thus it is not unique.

Q3

Part A

We can use graphs to solve the problem. First, construct a graph where the vertices are individual actors from the list and edges connect actors who appeared in a shared movie. Then, we just need to find the shortest path from the vertex for Kevin Bacon and the actor in question. The length of this path is an actor's Bacon number. This is a classic unweighted shortest path problem.

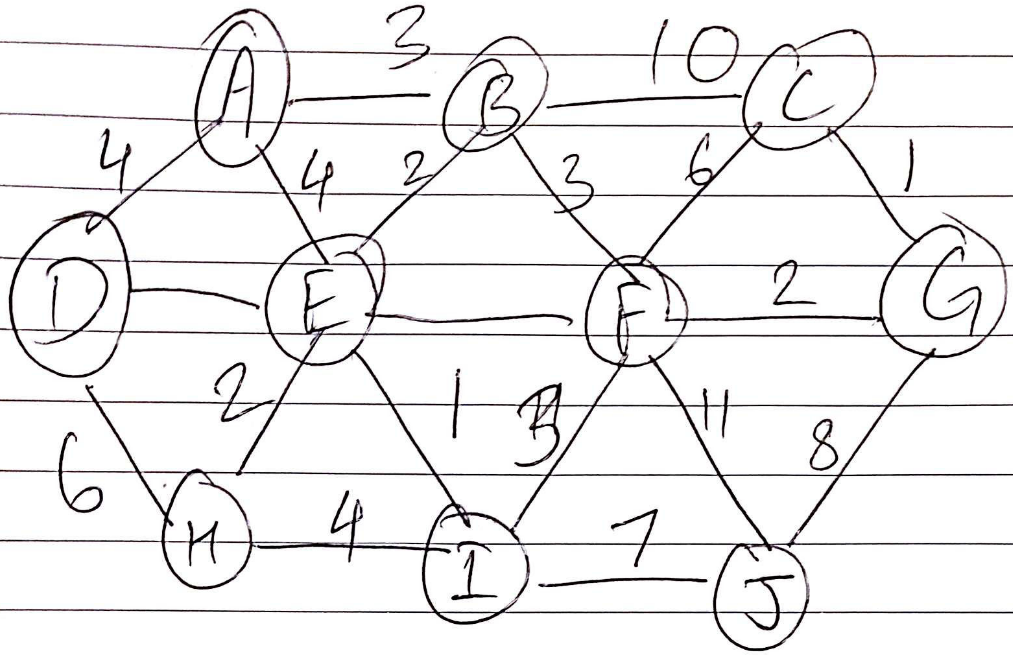
This can be solved using the algorithm described in section 9.3.1 of the textbook. Each actor stores a data value for distance (from Kevin Bacon) initialized to infinity. We start from Kevin Bacon and iterating through all vertices using BFS, we change the value of distance for all "unencountered" vertices adjacent to the current vertex . The value is changed to the value of the current vertex plus 1. When we reach the vertex that we care about the value of the vertex.distance determined (once it changes from infinity) is the Kevin Bacon number.

Part B

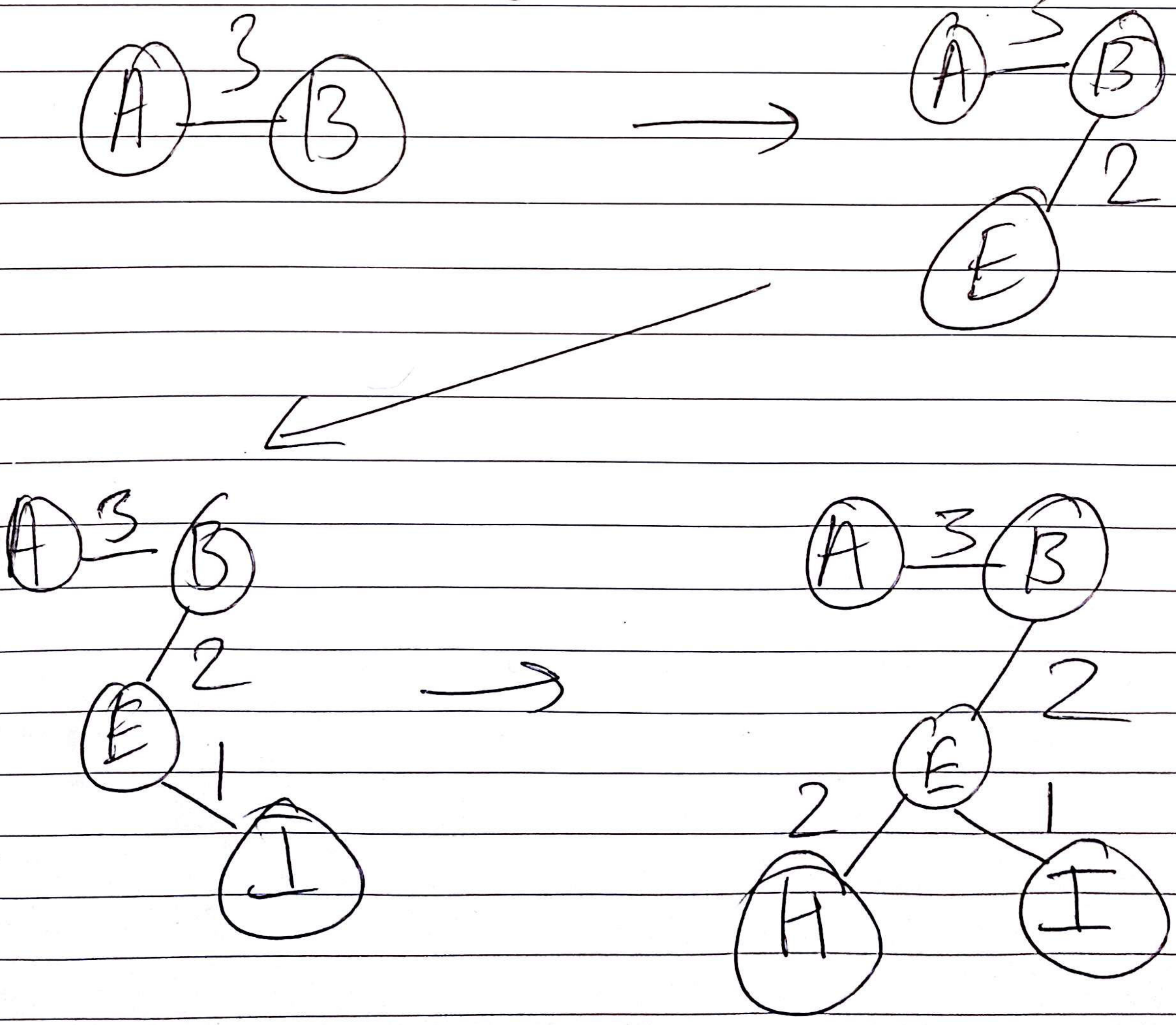
Using the same graph as in part a.) this question becomes a matter of finding the vertex with the *longest* shortest path from the vertex for Kevin Bacon. We assume that each vertex has an instance variable (v.dist) that equals the distance from Kevin Bacon. This is used in the algorithm for finding shortest paths. Then, this problem could be solved by iterating through the vertices (using for example DFS traversal) keeping track of a variables called maxBaconNumber and maxBaconActor that holds the largest distance value and the corresponding Actor's name encountered so far in the traversal.

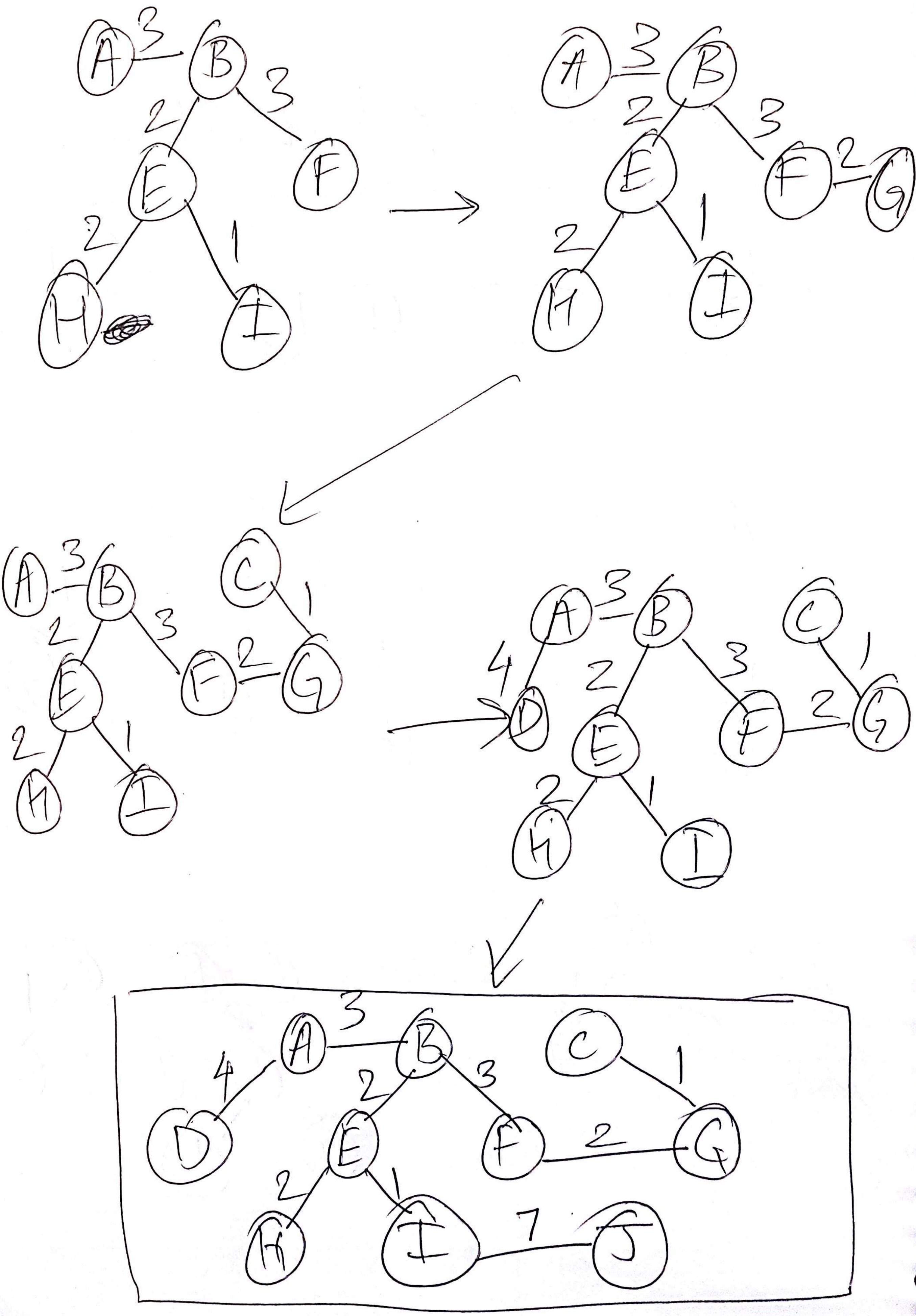
Part C

We would take one actor as a starting vertex and then perform the shortest path algorithm (from part a.) Thus the algorithm would be exactly the same. The only difference would be that instead of using Kevin Bacon as the starting vertex, we would use one of the two actors we are interested in. Then we would find the minimum path length from this actor to the other actor.

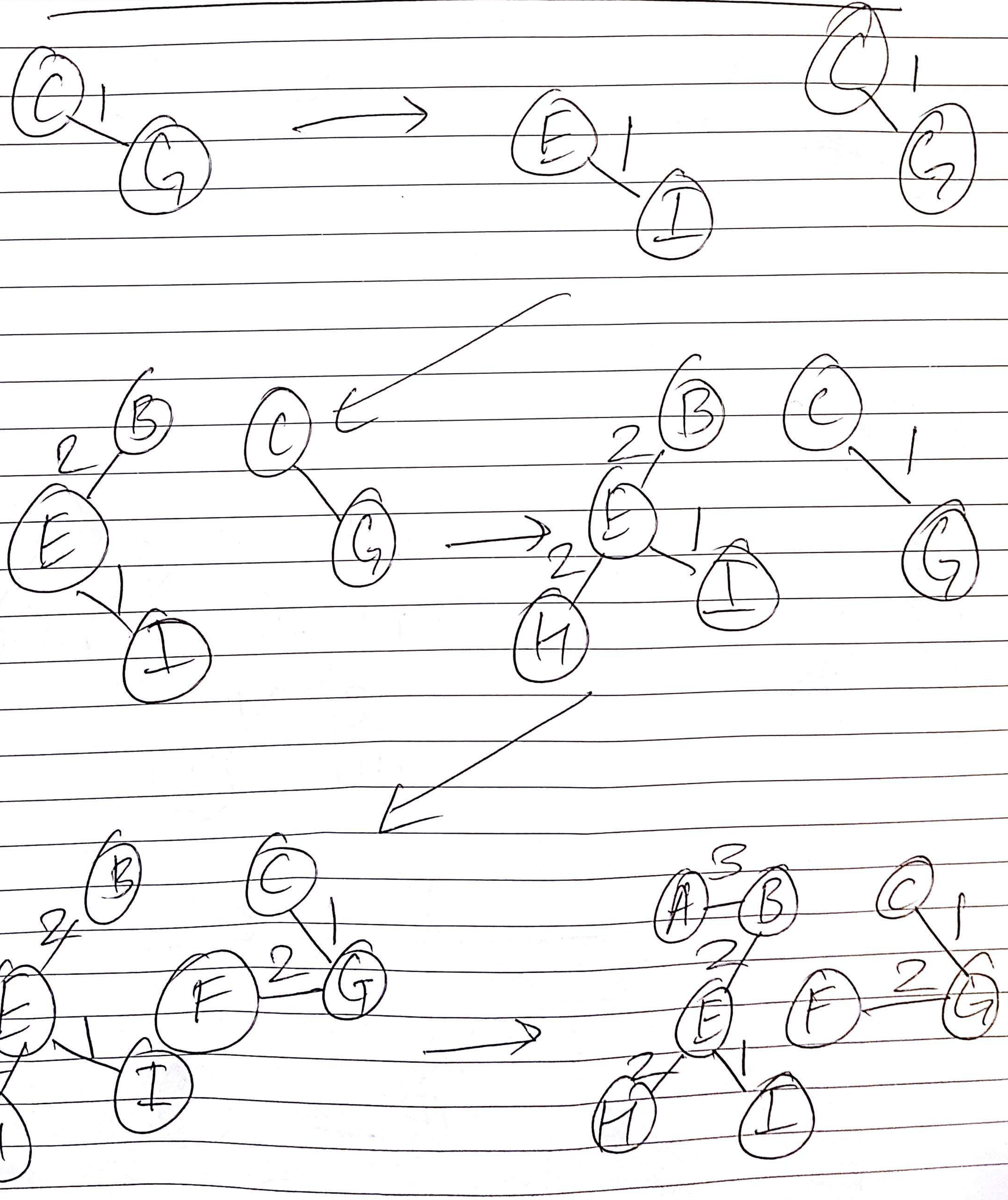


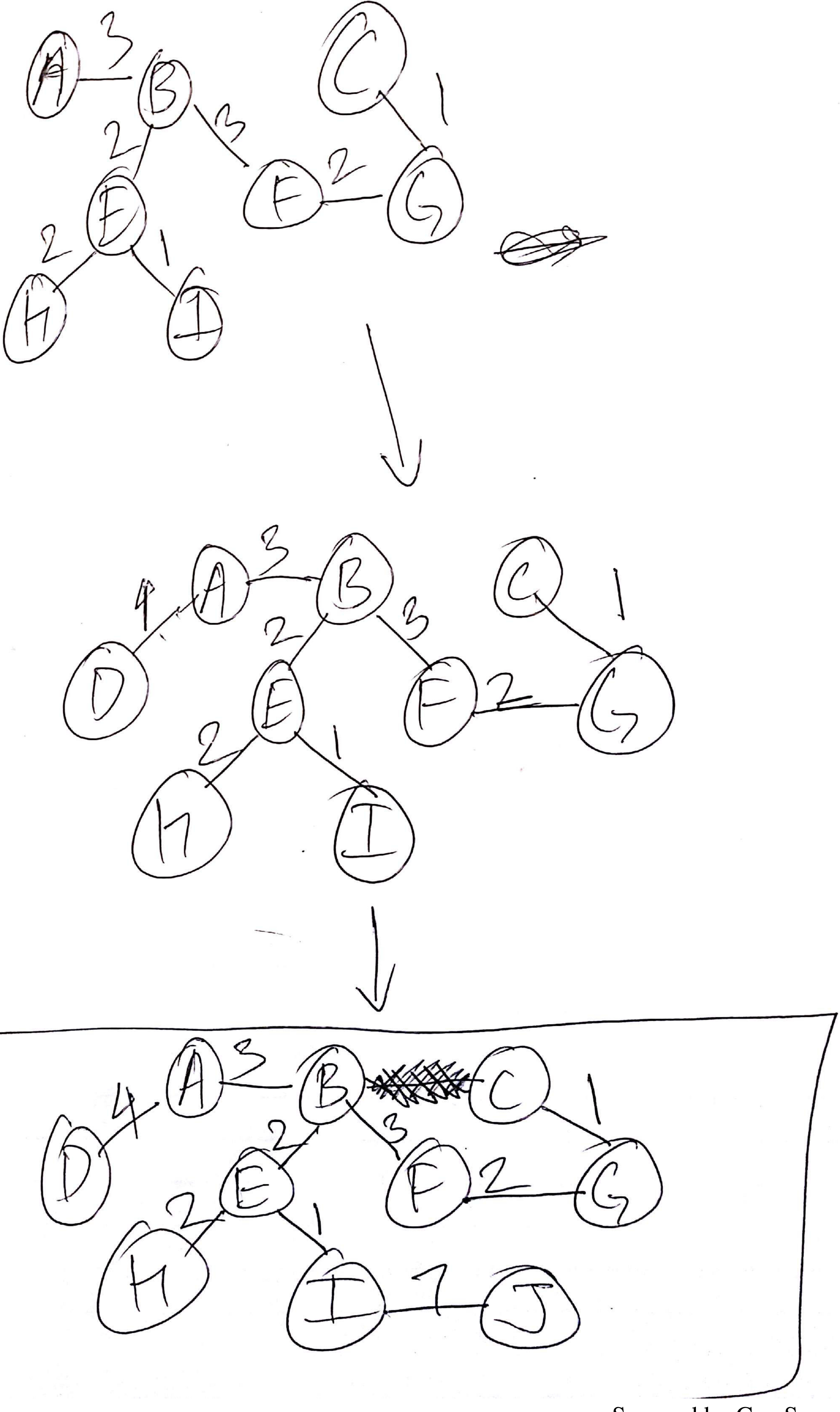
Prim's Algorithm




A
B
E
F
H
I
3
3
2
2
1
A
B
E
F
G
H
I
3
2
3
2
2
1
A
B
C
E
F
G
H
I
3
2
3
1
2
2
1
A
B
C
D
E
F
G
H
I
3
4
2
3
1
2
2
1
A
B
C
D
E
F
G
H
I
J
3
4
2
3
1
2
2
1
7

Krusal's Algorithm.




A
3
B
C
1
2
3
E
F
2
G
2
1
H
I
A
3
B
C
1
4
2
3
D
E
F
2
G
2
1
H
I
A
3
B
C
1
4
2
3
D
E
F
2
G
2
1
H
I
7
J