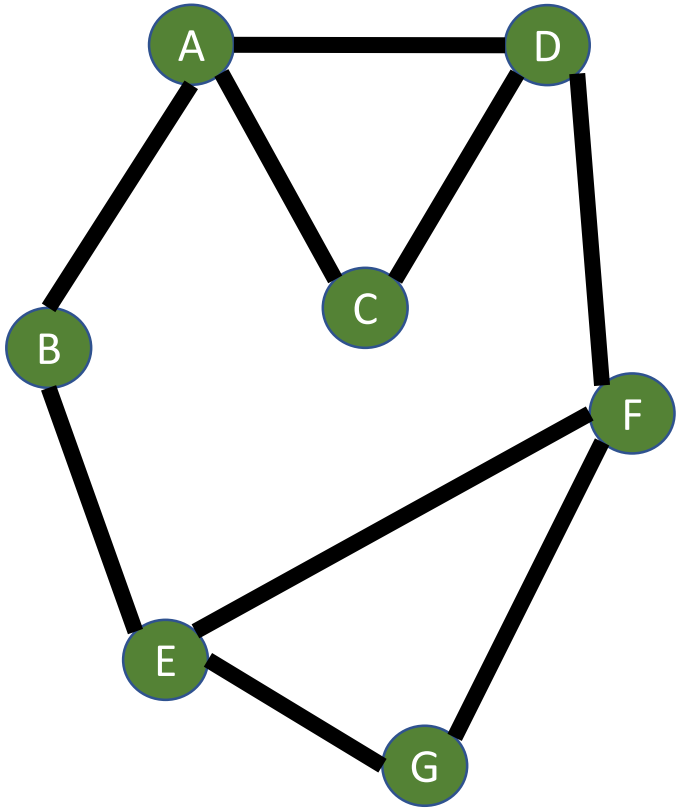
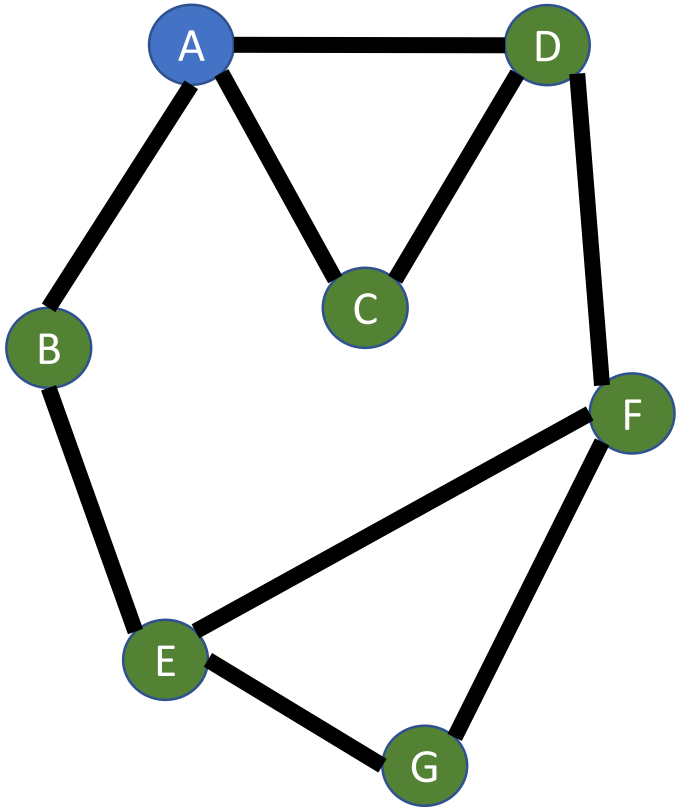
S = A



Q

```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A



Q

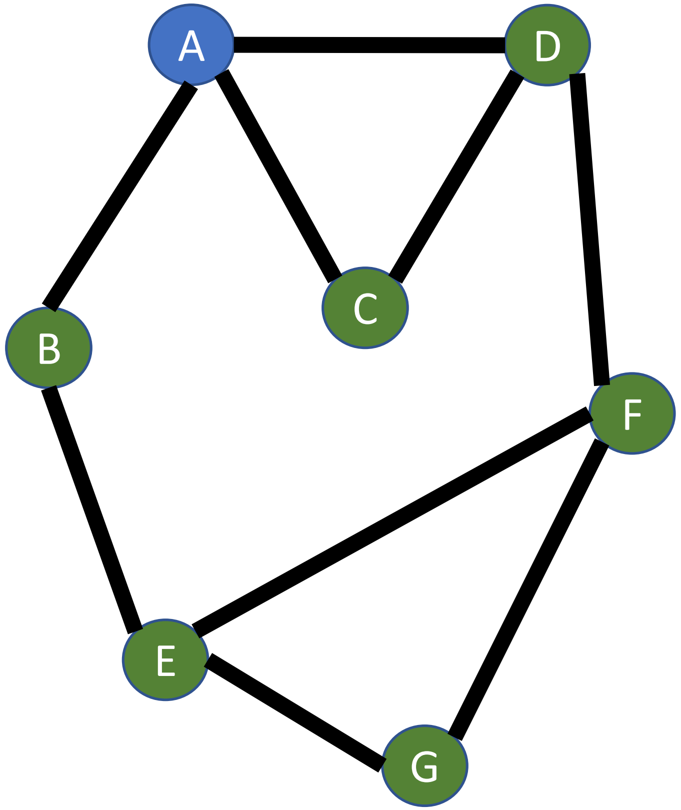
A

```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = A

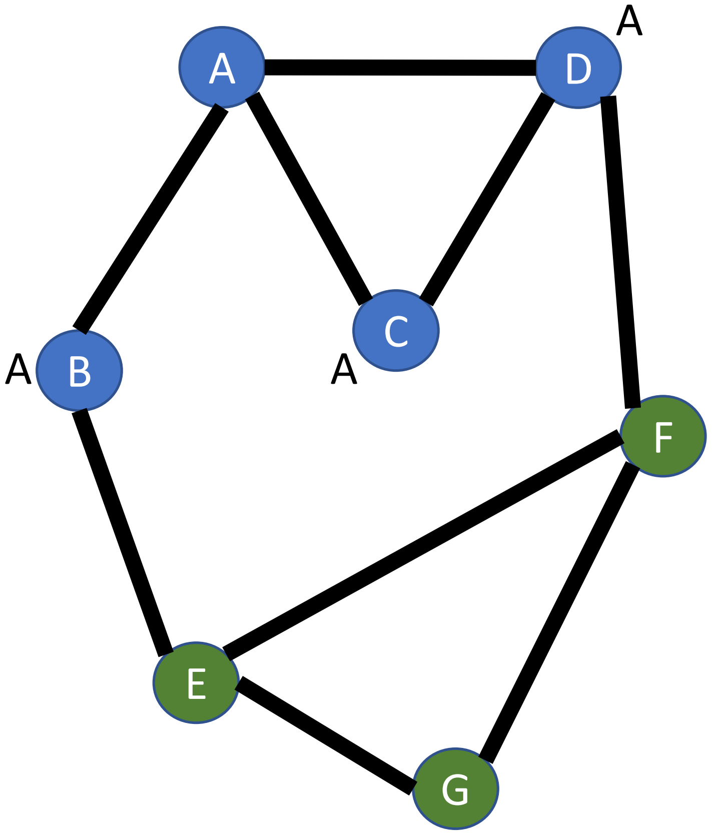
Q



```
Breadth-First-Search( Graph G=(V,E),  s in V )
    //Step 0: Mark s, put s into a queue
    mark s
    Q.enqueue(s)

    //Step 1: Enter BFS loop
    while( Q not empty )
        //Step 1.1: get item from Q
        x = Q.dequeue()

        //Step 1.2: visit all of x's
        //unvisited neighbors
        for each unmarked y in x.neighbors()
            mark y
            Q.enqueue(y)

            //Augment:
            y.bread = x
```

S = A

x = A



Q

B

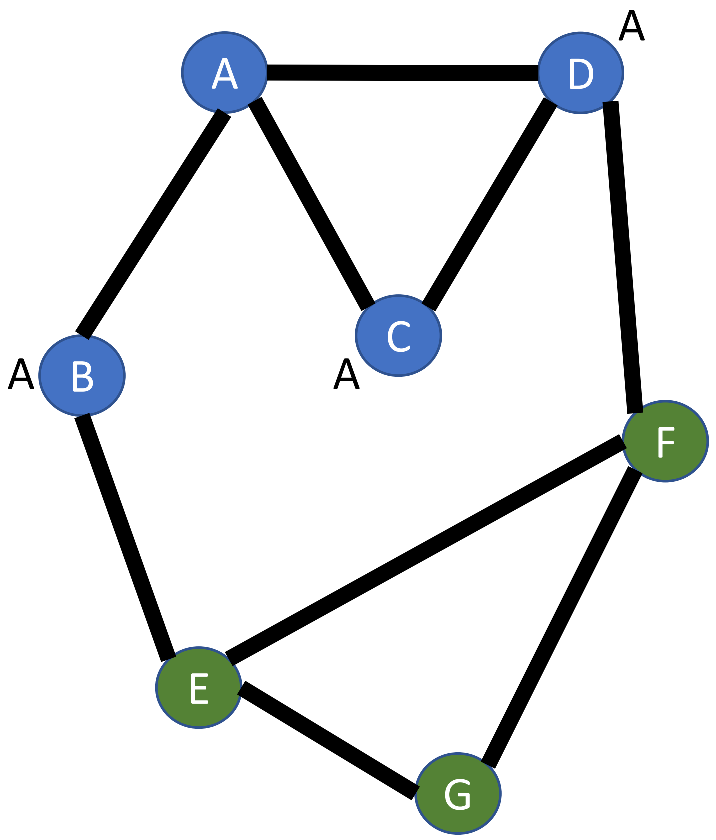
C

D

```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = ?



Q

B

C

D

```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

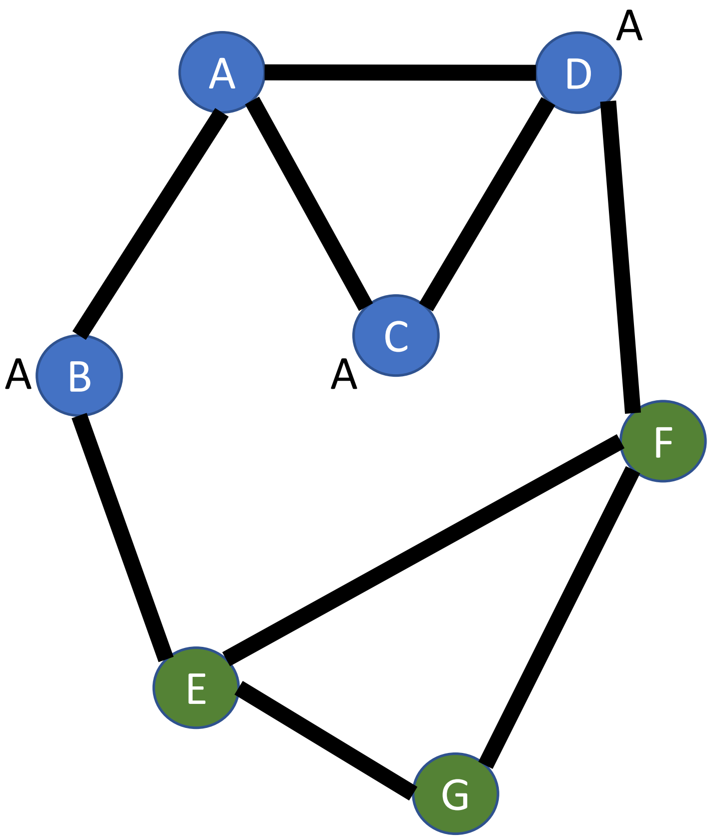
S = A

x = B

Q

C

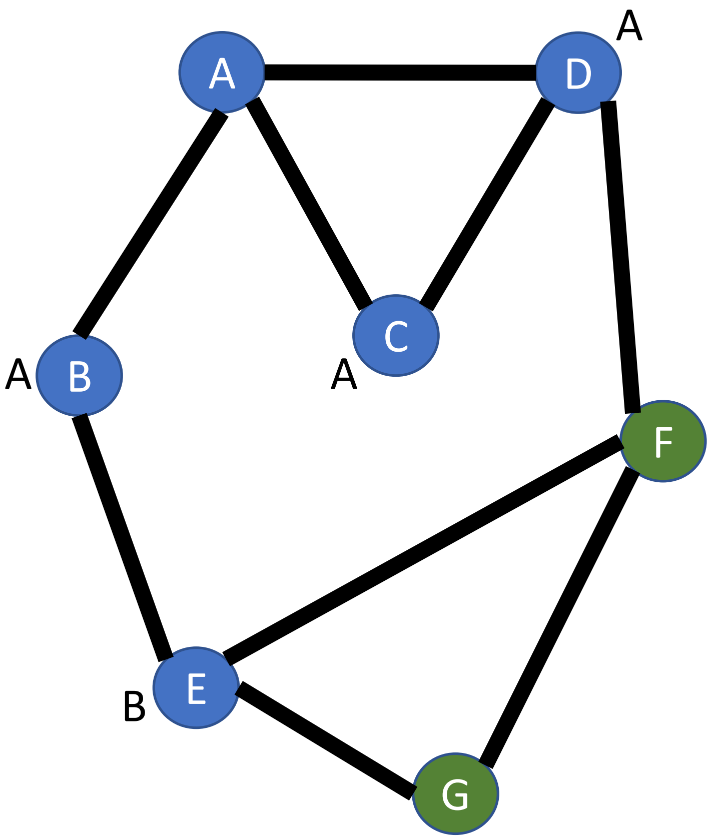
D



```
Breadth-First-Search( Graph G=(V,E),  s in V )
          //Step 0: Mark s, put s into a queue
          mark s
          Q.enqueue(s)

          //Step 1: Enter BFS loop
          while( Q not empty )
                    //Step 1.1: get item from Q
                    x = Q.dequeue()

                    //Step 1.2: visit all of x's
                    //unvisited neighbors
                    for each unmarked y in x.neighbors()
                              mark y
                              Q.enqueue(y)

                              //Augment:
                              y.bread = x
```

S = A

x = B



Q

C

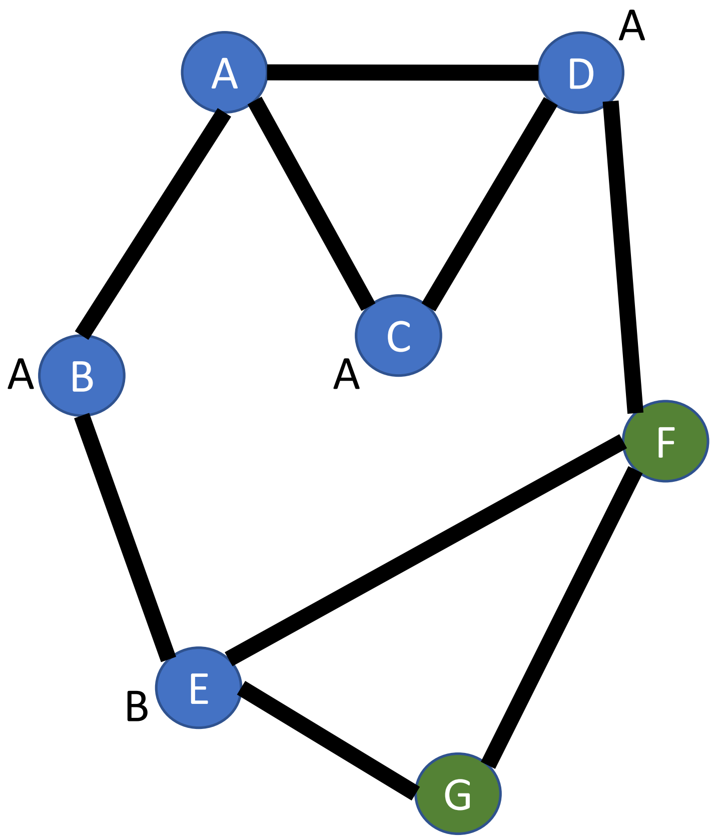
D

E

```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = ?



Q

C

D

E

```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

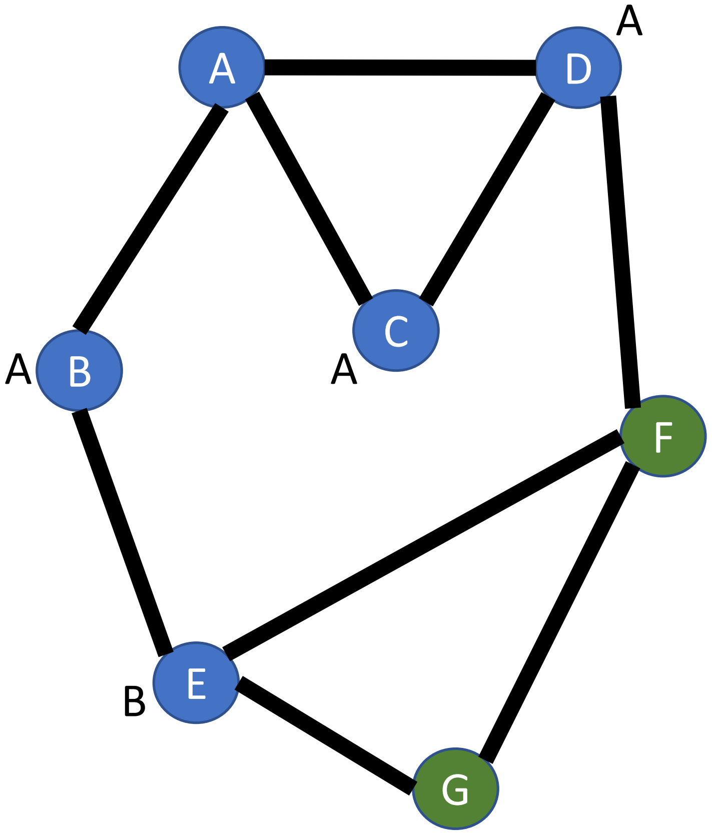
S = A

x = C

Q

D

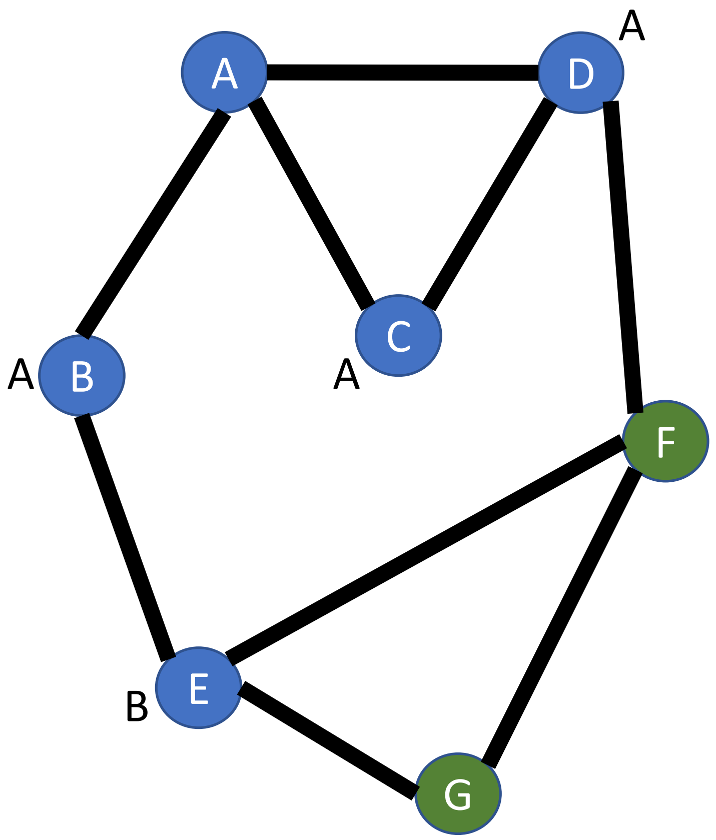
E



```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = ?



Q

D

E

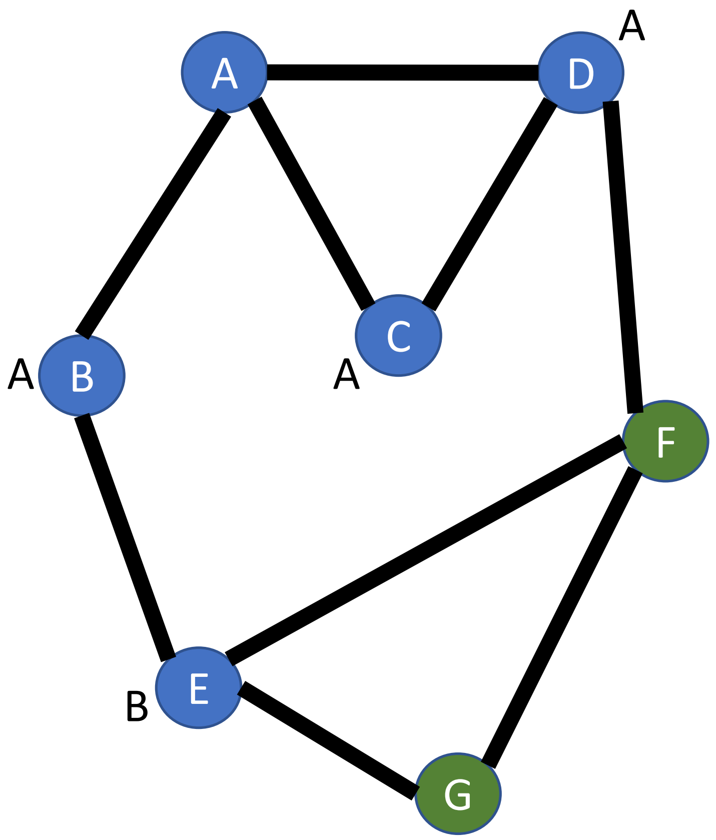
```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = D

Q

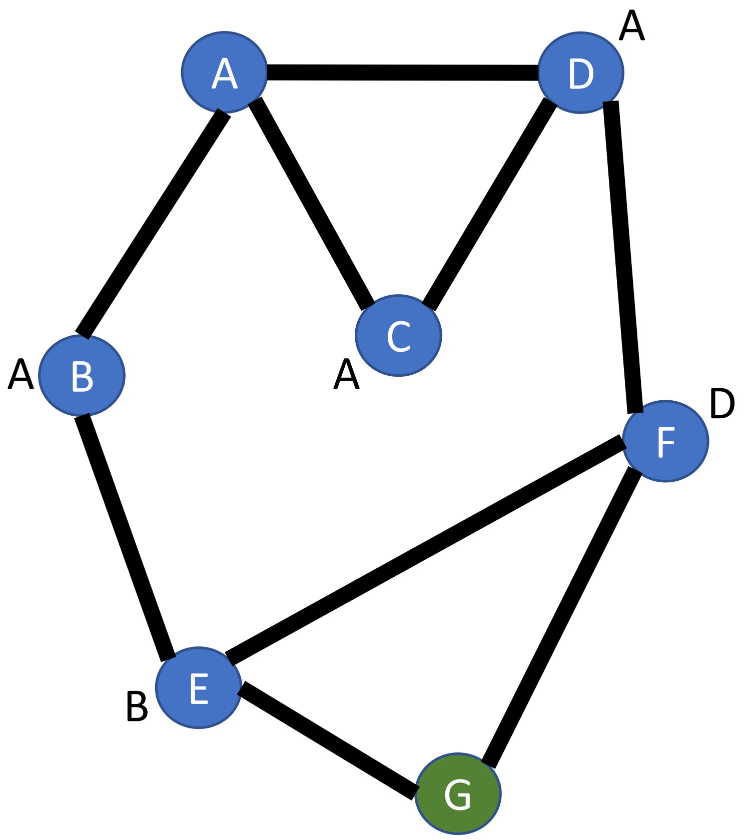
E



```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = D



Q

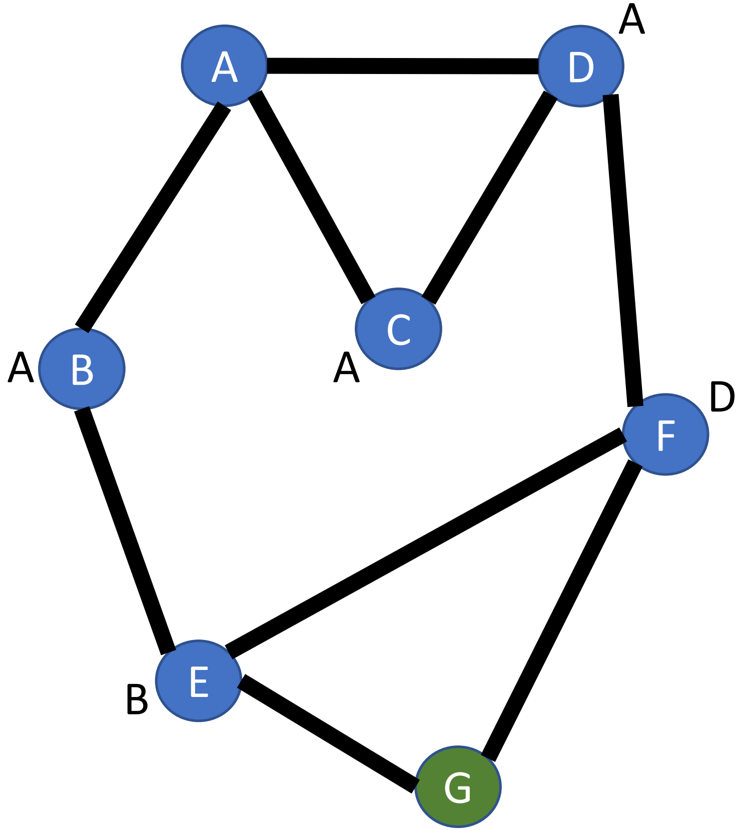
E

F

```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = ?



Q

E

F

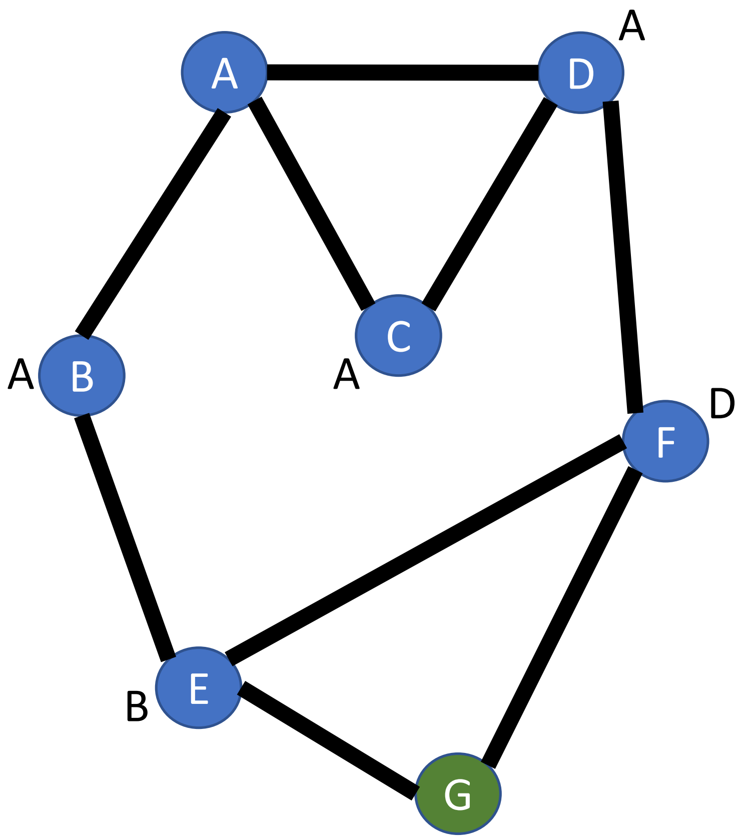
```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = E

Q

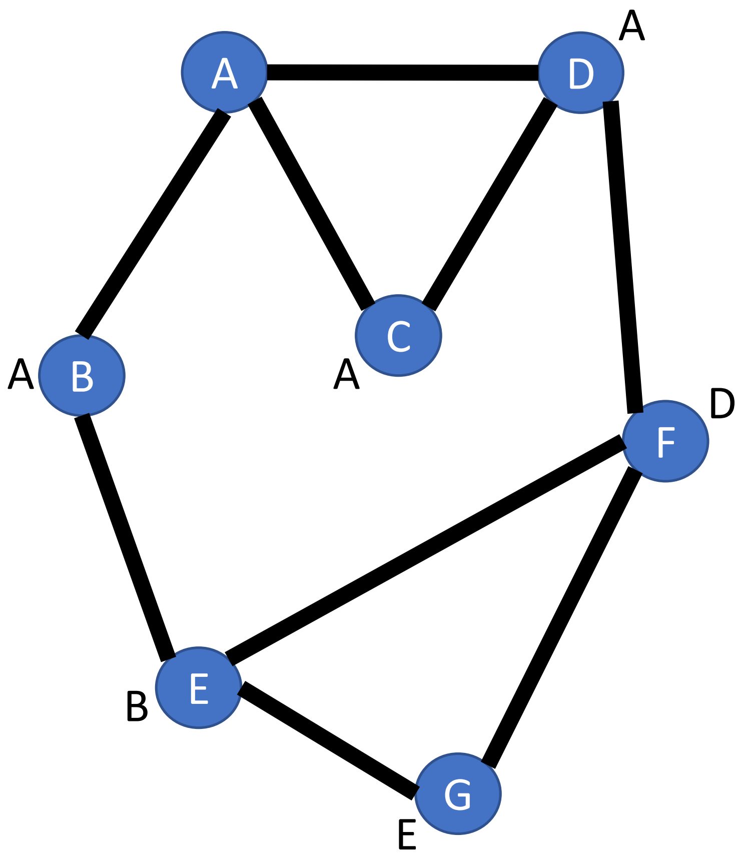
F



```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = E



Q

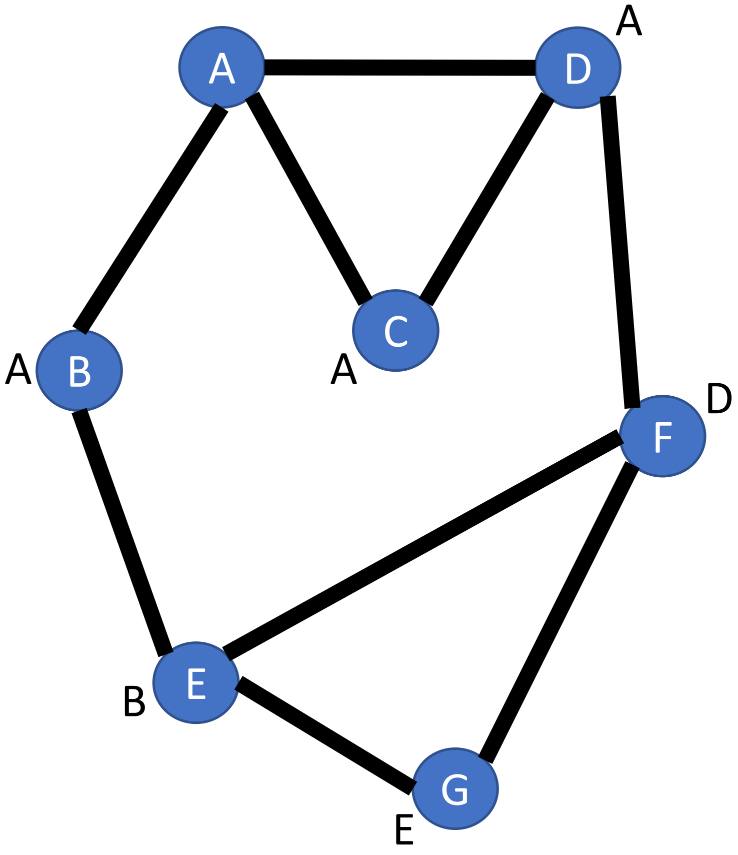
F

G

```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = ?



Q

F

G

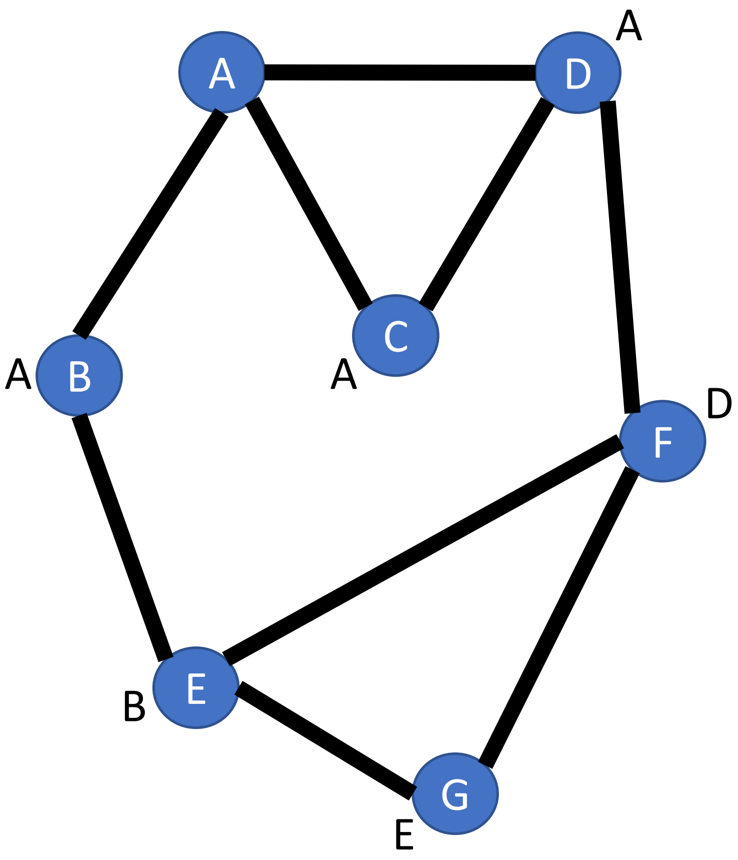
```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
            //Step 1.1: get item from Q
            x = Q.dequeue()

            //Step 1.2: visit all of x's
            //unvisited neighbors
            for each unmarked y in x.neighbors()
                mark y
                Q.enqueue(y)

                //Augment:
                y.bread = x
```

S = A

x = F

Q

G

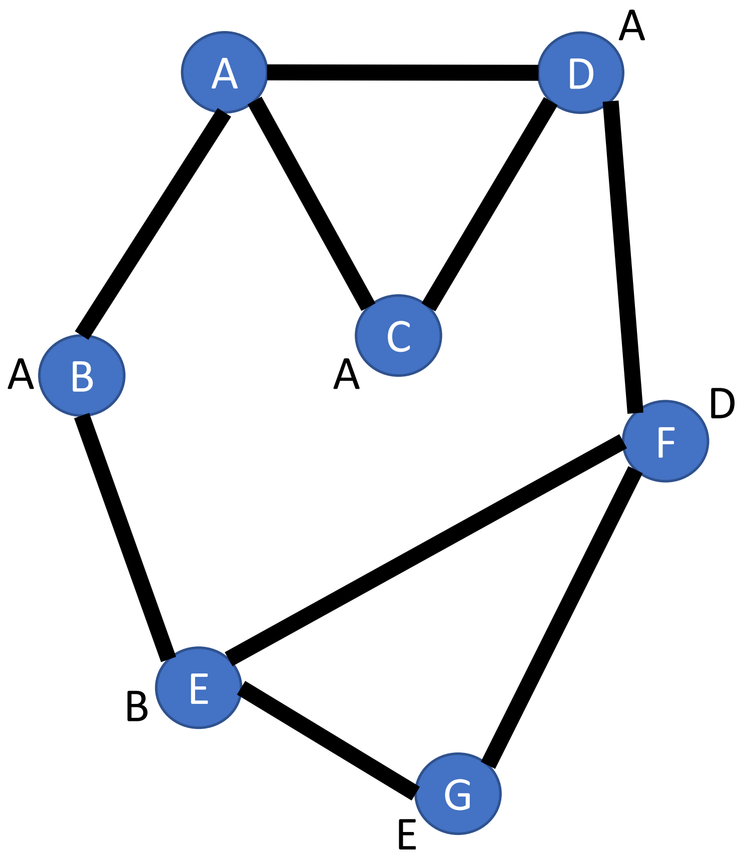


```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

x = G

Q

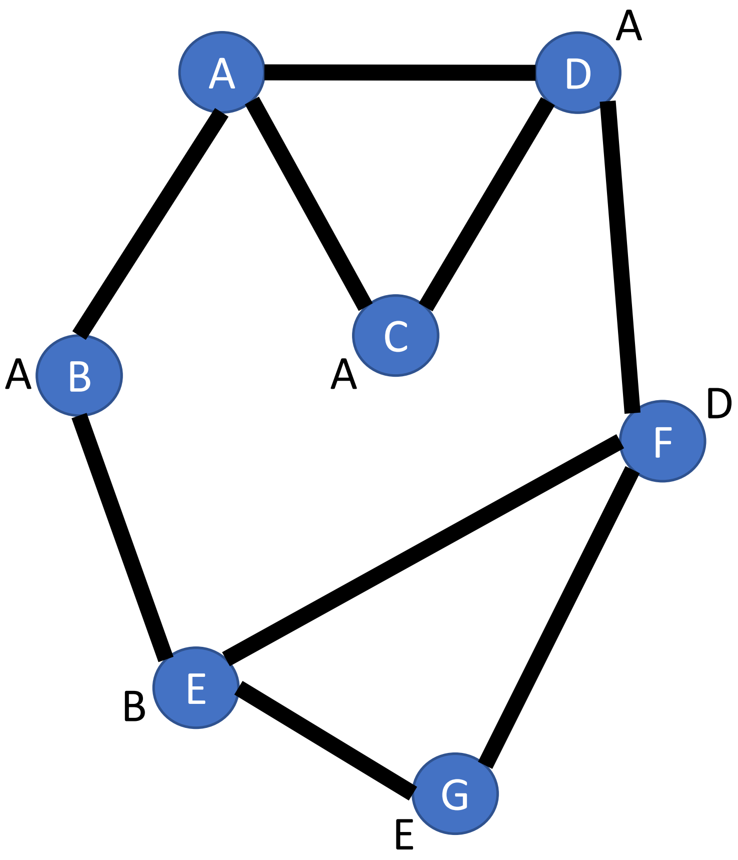


```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

S = A

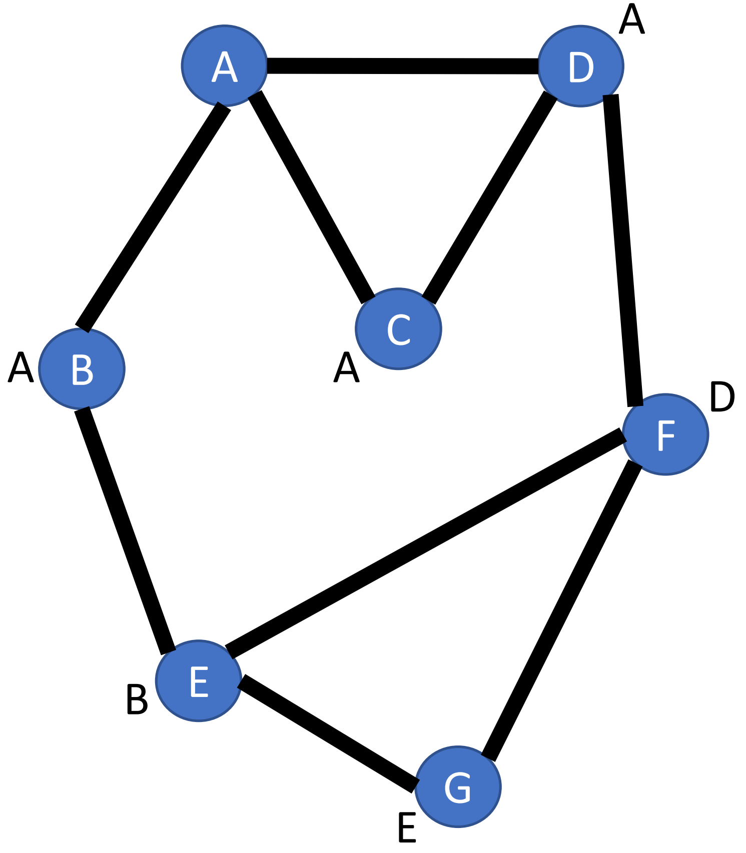
DONE

Q



```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
                x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)

                        //Augment:
                        y.bread = x
```

Run time? (In terms of |V| and |E|)

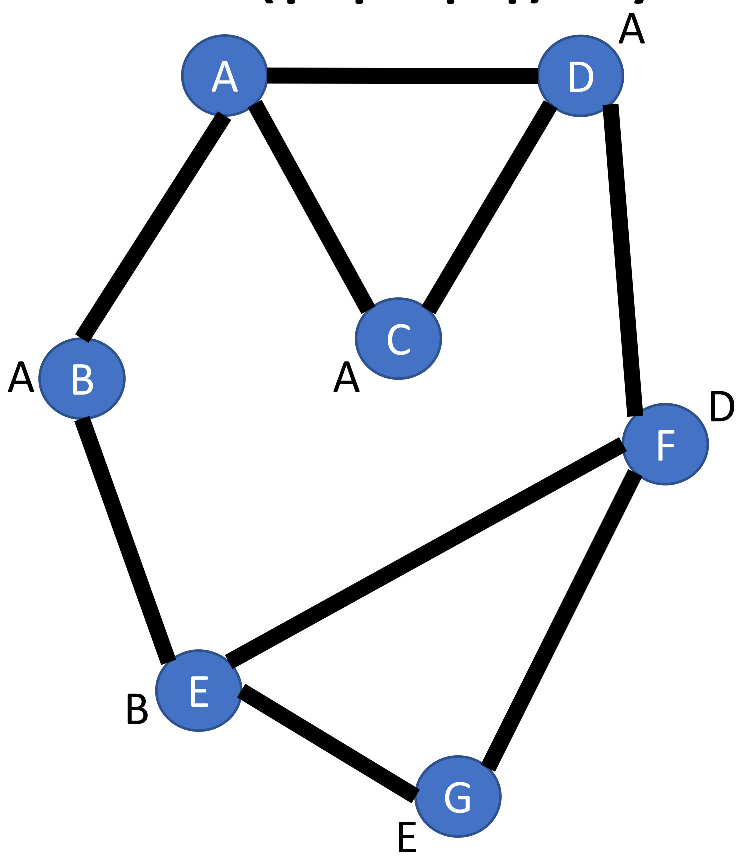


```
Breadth-First-Search( Graph G=(V,E),  s in V )
        //Step 0: Mark s, put s into a queue
        mark s
        Q.enqueue(s)

        //Step 1: Enter BFS loop
        while( Q not empty )
                //Step 1.1: get item from Q
  O(1)          x = Q.dequeue()

                //Step 1.2: visit all of x's
                //unvisited neighbors
                for each unmarked y in x.neighbors()
                        mark y
                        Q.enqueue(y)
            O(1)
                        //Augment:
                        y.bread = x
```

Run time?

O(|E|)

Or O(|V| + |E|) if you want to visit all vertices.



Breadth-First-Search(Graph G=(V,E), s in V)

```
    //Step 0: Mark s, put s into a queue
    mark s
    Q.enqueue(s)

    //Step 1: Enter BFS loop
    while( Q not empty )
        //Step 1.1: get item from Q
O(1)    x = Q.dequeue()

        //Step 1.2: visit all of x's
        //unvisited neighbors
        for each unmarked y in x.neighbors()
            mark y
            Q.enqueue(y)
O(1)
            //Augment:
            y.bread = x
```