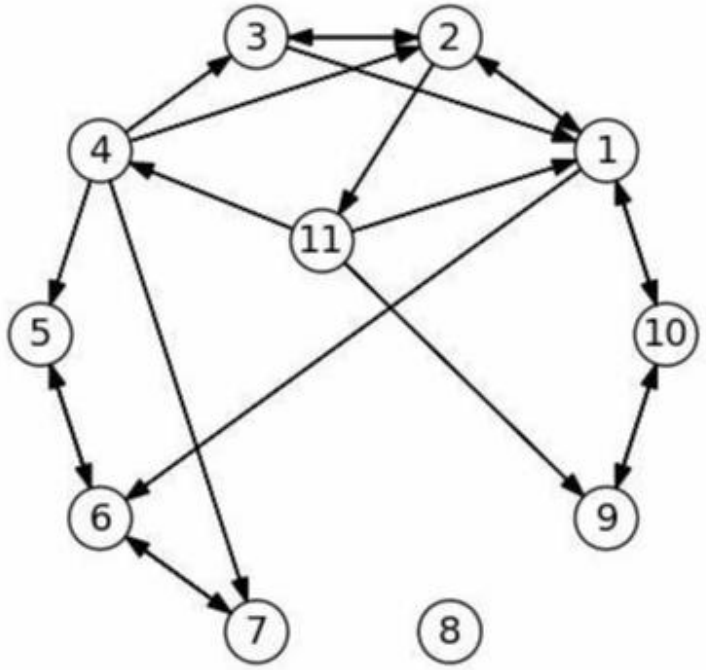
Apply DFS algorithm on the given graph and answer the following questions.

(**Note**: The adjacent nodes of a vertex are to be traversed in increasing order, and DFS algorithm will also work in **numerical order**.) **The source node is '11'**.

1. Write the type of each edge of the below graph, complete work should be there.
2. Write updated details for all the vertices when DFS-VISIT (G, u)
   [**7th line inside the DFS-VISIT**] called for the **6th time** (Just Called) write the values for all vertices ('**Parent'**, '**color'**, '**d-time', 'f-time')**.
3. Write down details for all the vertices when **Line 8** inside the DFS-VISIT completed its work for **5th time** - write the values of their ('**Parent'**, '**color'**, '**d-time', 'f-time')**.

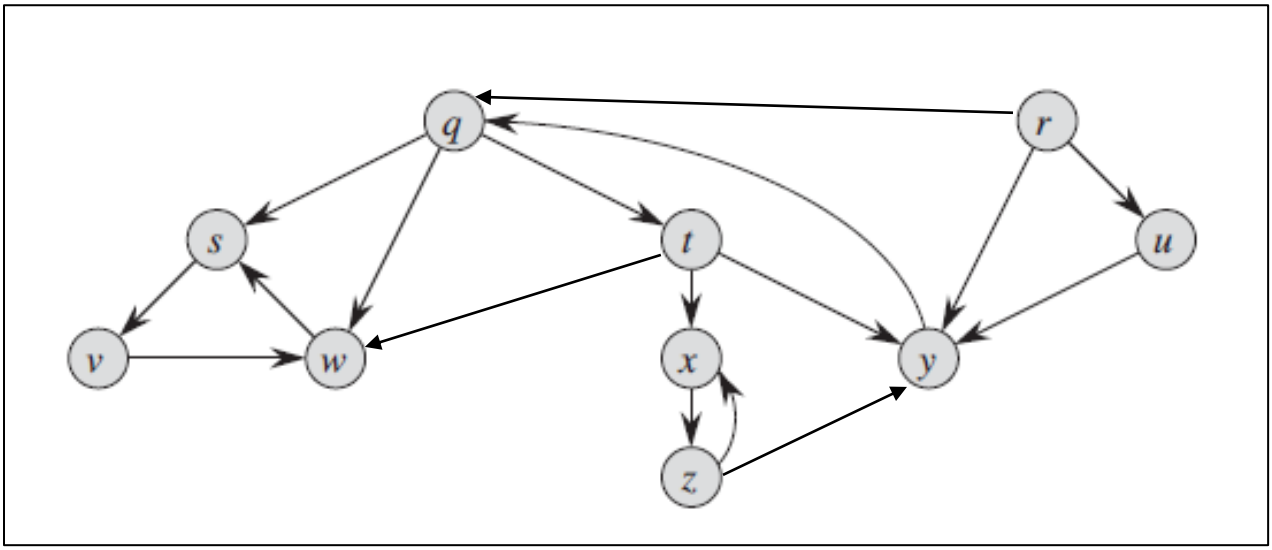
```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)

DFS-VISIT(G, u)
1  time = time + 1
2  u.d = time
3  u.color = GRAY
4  for each v ∈ G.Adj[u]
5      if v.color == WHITE
6          v.π = u
7          DFS-VISIT(G, v)
8  u.color = BLACK
9  time = time + 1
10 u.f = time
```



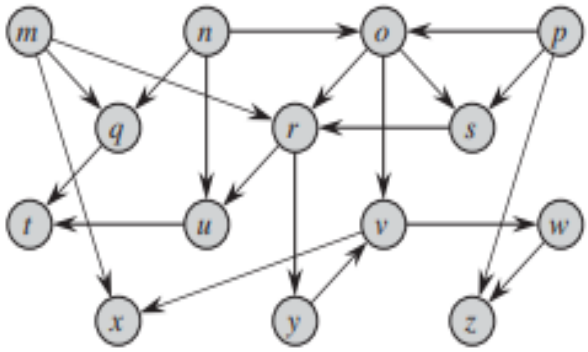
```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)

DFS-VISIT(G, u)
1  time = time + 1
2  u.d = time
3  u.color = GRAY
4  for each v ∈ G.Adj[u]
5      if v.color == WHITE
6          v.π = u
7          DFS-VISIT(G, v)
8  u.color = BLACK
9  time = time + 1
10 u.f = time
```

Apply DFS algorithm on the given graph and answer the following questions. (**Note**: The adjacent nodes of a vertex are to be traversed in alphabetical order, and DFS algorithm will also work in **alphabetical order**.) **The source node is 's'**.

1. Write the type of each edge of the below graph, complete work should be there.
2. Write updated details for all the vertices when DFS-VISIT (G, u) [**7th line inside the DFS-VISIT**] called for the **6th time** (Just Called) write the values for all vertices (**'Parent'**, **'color'**, **'d-time', 'f-time')**.
3. Write down details for all the vertices when **Line 8** of DFS-VISIT (G, u) completed its work for **5th time** - write the values of their (**'Parent'**, **'color'**, **'d-time', 'f-time')**.



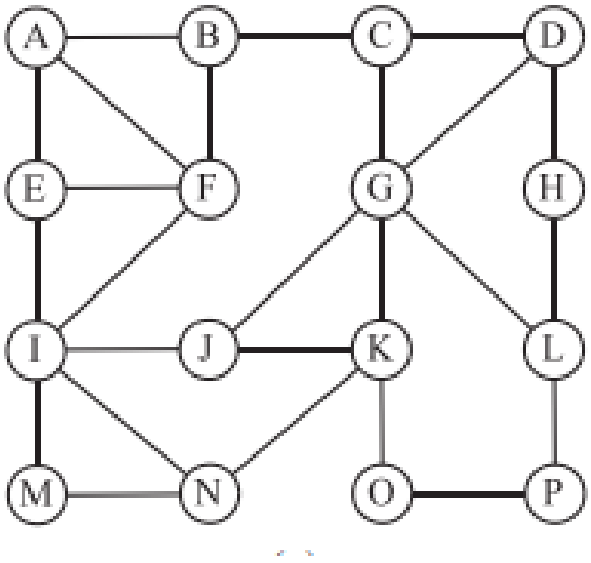
```
BFS(G, s)
1  for each vertex u ∈ G.V − {s}
2      u.color = WHITE
3      u.d = ∞
4      u.π = NIL
5  s.color = GRAY
6  s.d = 0
7  s.π = NIL
8  Q = ∅
9  ENQUEUE(Q, s)
10 while Q ≠ ∅
11     u = DEQUEUE(Q)
12     for each v ∈ G.Adj[u]
13         if v.color == WHITE
14             v.color = GRAY
15             v.d = u.d + 1
16             v.π = u
17             ENQUEUE(Q, v)
18     u.color = BLACK
```



Apply BFS algorithm on the given graph and answer the following questions. The source node is '**n**'. (**Note**: The adjacent nodes of a vertex are to be traversed in **alphabetical** order.)

1. Mark all the edges and draw the new graph with all the updated vertices and edges.
2. Write the names of the nodes/vertices into two sets, when L17 is reached for the 6th time and also write the values of their '**distance'**, '**predecessor'** and '**color'** attributes. Into two different sets. **Set1** [vertices available in the queue], & **Set2** [vertices dequeue from the queue]
3. Write the names of the nodes/vertices into two sets, when L13 is executed for the 12th time and also write the values of their '**distance'**, '**predecessor'** and '**color'** attributes. **Set1** [vertices available in the queue], & **Set2** [vertices dequeue from the queue]

```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```



Apply BFS algorithm on the given graph and answer the following questions. The source node is '**K**'. (**Note**: The adjacent nodes of a vertex are to be traversed in **alphabetical** order.)

1. Mark all the edges and draw the new graph with all the updated vertices and edges.
2. Write the names of the nodes/vertices into two sets, when L17 is reached for the 6th time and also write the values of their '**distance'**, '**predecessor'** and '**color'** attributes. Into two different sets. **Set1** [vertices available in the queue], & **Set2** [vertices dequeue from the queue]
3. Write the names of the nodes/vertices into two sets, when L13 is executed for the 12th time and also write the values of their '**distance'**, '**predecessor'** and '**color'** attributes. **Set1** [vertices available in the queue], & **Set2** [vertices dequeue from the queue]