Adjacency matrix

	A	B	C	D
A	0	1	1	0
B	1	0	1	0
C	1	1	0	1
D	0	0	1	0

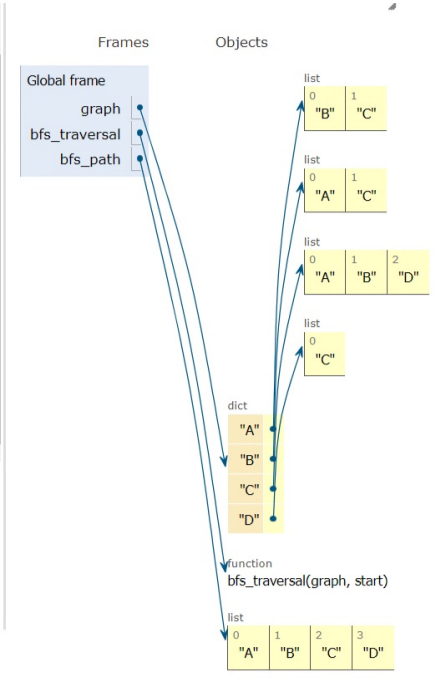
Adjacency list

A	B	C	
B	A	C	
C	A	B	D
D	C		

Edge list

0	A	B	A/B
1	A	C	A/C
2	B	C	B/C
3	C	D	C/D

code: https://onlinegdb.com/eKx3cVUimA



Adjacency matrix

	A	B	C	D	E	F
A	0	1	1	0	0	1
B	1	0	0	0	0	0
C	1	0	0	1	0	0
D	0	0	1	0	1	0
E	0	0	0	1	0	0
F	1	0	0	0	1	0

Adjacency list

A B C F

B A

C A D

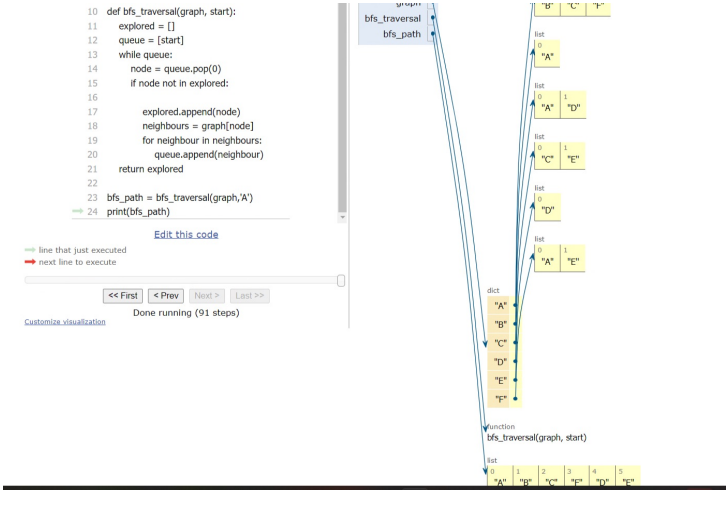
D C E

E D

F A E

Edge list

0	A	B	A/B
1	A	C	A/C
2	A	F	A/F
3	C	D	C/D
4	D	E	D/E
5	F	E	F/E

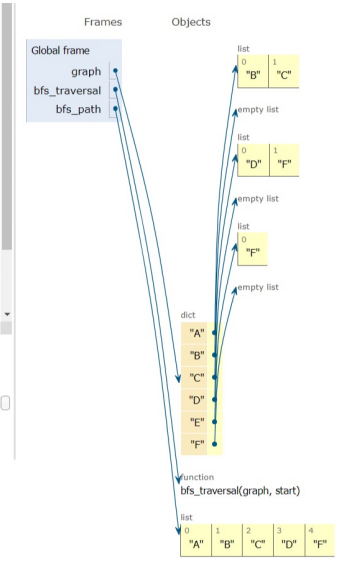


Code:

https://onlinegdb.com/-ap3l56sb8

Ex3.

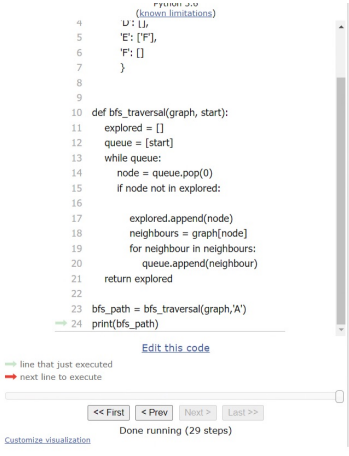
Create edge A,B,C,D,E,F and connect AB,AC,CD,CF,EF

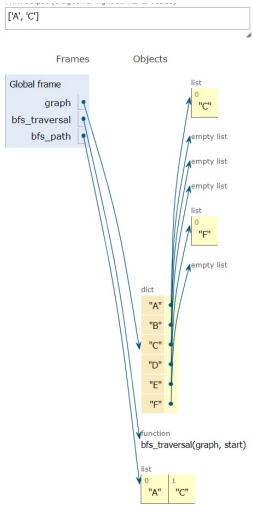


Code:

```
graph = {'A': ['B', 'C'],
         'B': [],
         'C': ['D', 'F'],
         'D': [],
         'E': ['F'],
         'F': []
         }


def bfs_traversal(graph, start):
    explored = []
    queue = [start]
    while queue:
        node = queue.pop(0)
        if node not in explored:

            explored.append(node)
            neighbours = graph[node]
            for neighbour in neighbours:
                queue.append(neighbour)
    return explored

bfs_path = bfs_traversal(graph,'A')
print(bfs_path)
```

Disconnect CF, AB, CD





Connect AE, BC, DF

```
5         'E': [],
6         'F': []
7         }
8
9
10  def bfs_traversal(graph, start):
11      explored = []
12      queue = [start]
13      while queue:
14          node = queue.pop(0)
15          if node not in explored:
16
17              explored.append(node)
18              neighbours = graph[node]
19              for neighbour in neighbours:
20                  queue.append(neighbour)
21      return explored
22
23  bfs_path = bfs_traversal(graph,'A')
24  print(bfs_path)
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Done running (29 steps)

Customize visualization