1. Tracing BFS

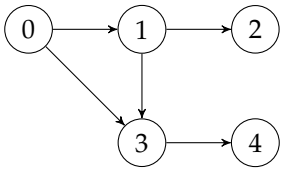
Breadth-first search (BFS) explores all of the neighbors of a node before moving on to nodes farther away.

The following code implements BFS using a queue:

```
from collections import deque

def bfs(G, start):
    visited = [False] * len(G)
    queue = deque([start])
    visited[start] = True
    while len(queue) > 0:
        node = queue.popleft()
        print(node)
        for neighbor in G[node]:
            if not visited[neighbor]:
                visited[neighbor] = True
                queue.append(neighbor)
```

Consider the following graph:



The neighbors of each node are explored in the order shown below:

`0: [1, 3], 1: [2, 3], 2: [], 3: [4], 4: []`

a. Write the order in which the nodes are printed.

[BFS order: _____ → _____ → _____ → _____ → _____]

Solution: 0 1 3 2 4

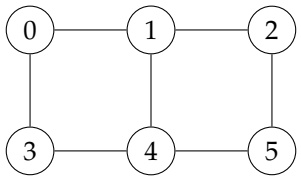
b. When BFS visits node 0, which nodes are added to the queue? _____

Solution: 1 3

2. BFS Finds the Shortest Path

BFS explores a graph one **layer** at a time. The starting node is in the zeroth layer. Each following layer contains the nodes that are one more edge away from the starting node.

Consider the following graph:



We start BFS at node 0.

a. Write the nodes in each BFS layer.

Layer 0: _____

Solution: 0

Layer 1: _____ **Solution:** 1 3

Layer 2: _____ **Solution:** 2 4

Layer 3: _____ **Solution:** 5

b. What is the shortest number of edges from node 0 to node 5? _____

Solution: 3

c. Which of the following is a shortest path from node 0 to node 5?

(A) $0 \to 1 \to 2 \to 5$

(B) $0 \to 3 \to 4 \to 5$

(C) Both (A) and (B)

(D) Neither (A) nor (B)

Solution: C) Both (A) and (B)

d. Suppose BFS first reaches node 5 in Layer 3. Could there be a path from 0 to 5 using fewer than 3 edges?

(A) Yes

(B) No

Solution: (B) No