Breadth-First Search (BFS)

This document contains summary of some of the items discussed as part of the COMS 3110 Lectures. This document **does not replace the lecture materials**. This document may contain some topics that are not covered as part of the lecture; you will not be tested on those parts, they are made available to you for gaining further knowledge on topics/concepts that are related to class-lecture.

1 Topics Covered

- BFS
- Applications of BFS: connected components, shortest path, bipartite graph

2 Breadth-First Search (BFS)

- One may want to know if a particular vertex can be reached.
- Look at all the neighbors of a given node that can be reached by an edge.
- Repeat this over and over again to find out all the nodes that can be reached from a given node.

Properties of BFS:

- If u can be reached from v, then u must belong to some layer resulting from BFS starting from v.
- If u is in layer i resulting from BFS starting from v, then u is i steps away from v.
- For any BFS, the maximum number of layers is $|V| - 1$.

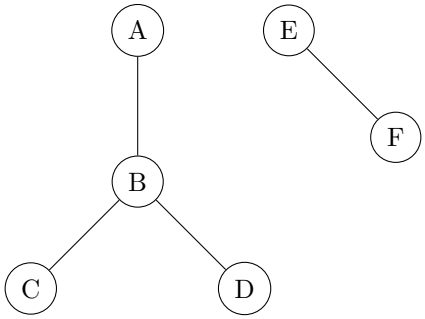
BFS pseudocode:

```
# Initially x.explored = false for each node x
BFS(v):
    v.explored = true
    v.layer = 0
    Initialize an empty queue Q
    Add v to Q (to the tail)
    while Q is not empty:
        Pop u from head of Q
        for all neighbors w of u:
            if w.explored == false then:
                add w to tail of Q
                w.explored = true
                w.layer = u.layer + 1
```

BFS runtime:

- Each vertex in the graph is added to the queue at most once.
- All the edges in the undirected graph are checked at most twice. Consider an edge (a, b), it is checked once when $u = a$ and once when $u = b$ (the u variable shown in the algorithm) in the algorithm.
- As a result, the runtime of BFS is $O(|V| + 2|E|) = O(|V| + |E|)$.

3 Connected Components



There are two connected components in the above graph. A, B, C, D is part of one connected component and E, F is part of the other. We know that calling BFS from a vertex v gives all vertices that are reachable from v, so we can use this fact to generate the connected components.

To Find Connected Components:

```
INPUT: an undirected graph G

Initialize v.explored = false and v.component = -1 for all vertices v in the graph G
i = 0
for all vertices v in G:
    if v.explored == false then:
        BFS(v, i)
        i = i + 1
```

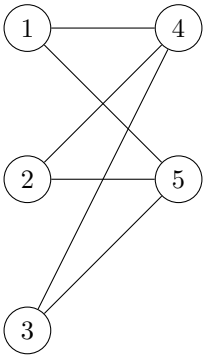
In this case, $BFS(v, i)$ is just a simple modification of the original BFS algorithm. Here, the BFS algorithm takes on an additional argument i to denote the component of all the vertices that are reachable from v.

Runtime:

- The runtime is still $O(|V| + |E|)$ because we are basically running BFS for i times, where i is the number of connected components in the graph. Essentially, we are still adding all the vertices in the graph to the queue once, and traversing each edge of the graph twice.

4 Bipartite Graph

A graph $G = (V, E)$ is a bipartite if V can be partitioned into A and B (i.e., $A \cap B = \{\}$, $A \cup B = V$) such that for all edge (u, v), $u \in A$, $v \in B$.



The above graph is bipartite, where $A = \{1, 2, 3\}$ and $B = \{4, 5\}$.

Observation 1: If a graph contains a cycle with an odd number of edges, it cannot be bipartite.

Observation 2: If L_0, L_1, L_2,... are the layers resulting from BFS exploration of a graph, then there is no edge between vertices in the same layer if an only if the graph is bipartite. If there is an edge between vertices in the same layer, that would imply the graph is not bipartite.

As a result, if we want to test whether a graph is bipartite or not, we can run BFS exploration on the graph and check if there exists an edge between vertices of the same layer. If there exists an edge between vertices in the same layer, the graph is not bipartite.