CSE421: Design and Analysis of Algorithms	April 10, 2019
Lecturer: Shayan Oveis Gharan	Lecture BFS and Connected Components

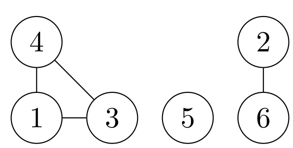
In class we discussed a pseudo-code of BFS(s); Here I have modified the code to maintain the level of each vertex in the BFS tree, in other words, the array L[] will have the shortest path distance from s to u for any vertex u in the connected component of s.

```
Function BFS(s)
    Initialize: mark all vertices "undiscovered'
    mark s "discovered"
    queue = { s }
    L[s]=0
    while queue not empty do
        u = remove_first(queue)
        for each edge {u, x} do
            if x is undiscovered then
                mark x discovered
                append x on queue
                L[x]=L[u]+1
            end
        end
        mark u fully-explored
    end
```

Algorithm 1: Computes the shortest path distance from s

Next, we write a code to determine the connected components of a graph. When we call the function Connected-Components, it will construct an array A such that for all vertices v in the same connected component $A[v]$ is the same.

For example, consider the following graph; it has 3 connected components: $\{1, 3, 4\}, \{5\}, \{2, 6\}$. If we run the code on the following graph, we are going to make 3 BFS calls:



1) First we call BFS(1) which visits vertices 1,3,4 and so we get $A[1] = A[3] = A[4] = 1$.

2) Then we call BFS(2) which visits vertices 2,6 and so $A[2] = A[6] = 2$.

3) Then we call BFS(5) which visits the vertex 5 and so we get $A[5] = 3$.

Note that we are not going to call BFS(3), BFS(4) and BFS(6). Because by the time the main loop gets to vertices 3, 4, and 6 they are already fully-explored.

```
Function BFS(s,c)
    mark s "discovered"
    queue = { s }
    A[s]=c
    while queue not empty do
        u = remove_first(queue)
        for each edge {u, x} do
            if x is undiscovered then
                mark x discovered
                append x on queue
                A[x]=c;
            end
        end
        mark u fully-explored
    end
Function Connected-Components
    Initialize: mark all vertices "undiscovered" and set c = 1
    for v = 1 → n do
        if v is undiscovered then
            BFS(v,c)
            c=c+1
        end
    end
```

Algorithm 2: Computes the Connected Components of a Graph

Also, observe that after running this code, for any pair of vertices u, v, there is a path connecting u to v in G if and only if $A[u] = A[v]$.