Revisiting BFS:

BFS(s):   Queue: FIFO

Discovered [s] = True

For all $v \in V \setminus \{s\}$:

  Discovered [v] = False.

~~L[0] = {s} // list~~ } L ← empty queue

~~$i \leftarrow 0$~~

$T \leftarrow \phi$ // Empty tree.

~~While L[i] is not empty:~~   While L is not empty:

  ~~L[i+1] ← [] // empty list~~ $u \leftarrow$ L.~~pop~~(u)

  ~~For each $u \in L[i]$:~~

    For each edge $(u,v) \in E$ incident on u:

      If Discovered [v] == false:

        Discovered [v] ← True

        $T \leftarrow T \cup \{(u,v)\}$

        ~~L[i+1].append (v)~~   L.append (v).

  ~~$i \leftarrow i+1$.~~

---

n - layers ⇒ ≤ n iterations of while.

$\sum_{v \in V} |N(v)| = 2m$.

O(m+n) running time.

Revisiting DFS:

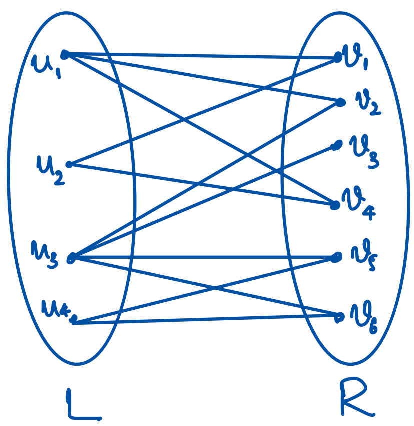
```
Stack S ← ⊥. // empty stack
DFS(s):
    S.push(s).

    While S is not empty:
        u ← S.pop()
        If Explored[u] == False :
            Explored[u] ← True
            For each (u,v) ∈ E :
                S.push(v).
```

```
R ← {}
DFS(u):
    Explored[u] ← True
    R ← R ∪ {u}
    For each (u,v) ∈ E :
        If Explored[v] == False:
            DFS(v) .
```

DFS(s)
Stack: { Last In First Out }

Applications: Testing bipartiteness.

Biparttite graphs: (L, R, E)



Lemma: A graph is bipartite if and only if it contains no odd cycles.

Pf: ($\Rightarrow$) A graph is bipartite $\Rightarrow$ no odd cycles.

$$\begin{array}{cccccc} A & B & A & & & B \\ | & | & & & & | \\ v_1, & v_2, & \ldots, & v_k, & & v_1 \end{array}$$ be a cycle.

$$\begin{array}{cccc} v_1 & v_2 & v_3 & v_1 \\ | & | & | & | \\ A & B & A & B \end{array}$$

($\Leftarrow$) No odd