→ Connected graph (Undirected) : For any pair of vertices $u, v \in V$ then $\exists$ path from u to v.

BFS on a particular vertex:

BFS(s):

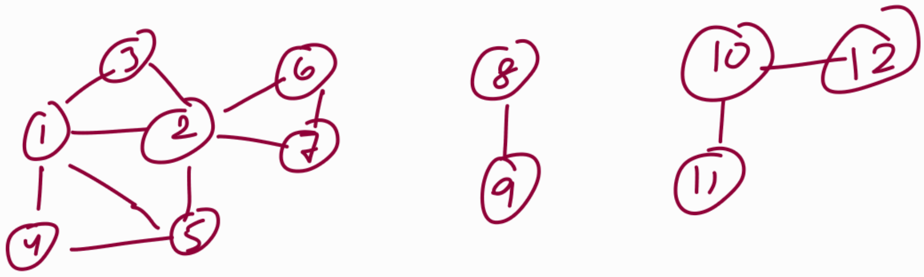
Discovered[s] = True
For all $v \in V \setminus \{s\}$
  Discovered[v] = False
} Init.

$L[0] \leftarrow \{s\}$
$i \leftarrow 0$
$T \leftarrow \phi$

While L[i] is not empty:
  $L[i+1] \leftarrow [\ ]$ } Init.
  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).

→ BFS used to check connectivity.
i.e., to check if u & v are connected, start BFS on u & search for v.

• Compute connected components.



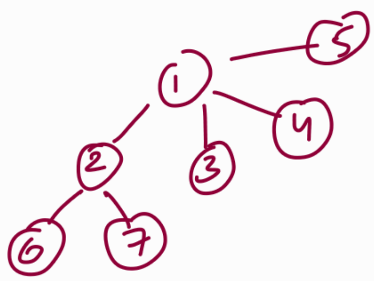
Init: For all $v \in V$

Discovered$[v] \leftarrow$ False.

Pick a vertex (Eg u):

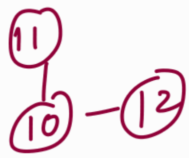
Run BFS(u) ← This gives connected comp. containing u.

BFS(1):



BFS(8):

BFS(11):



→ Book keeping: Component number.

→ BFS can be used to determine connected component.

• Testing bipartiteness

Bipartite graphs ≡ 2 colourable graph

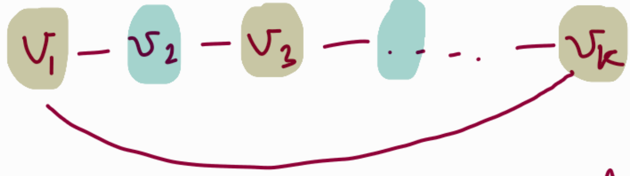
Lemma: A graph (UNDIRECTED) is bipartite iff it has no odd cycles.

Cycles with odd no. of edges.

Proof:

1) Bipartite ⇒ No odd cycle.

Suppose not. Suppose we have an odd cycle & it is 2-colourable.



k : Odd

v_1 & v_k have the same colour (∵ every odd vertex has the same colour)
i.e., (v_1, v_k) edge is monochromatic.

$\Rightarrow$ Not 2 colourable graph.

Contradiction.

2) No odd cycle $\Rightarrow$ Bipartite

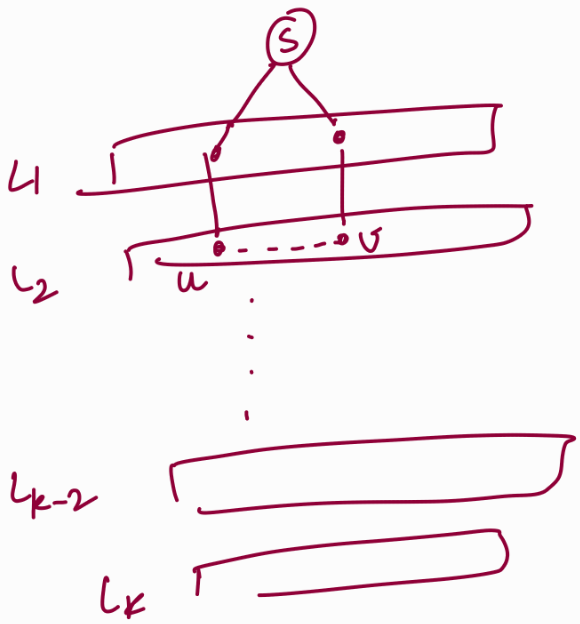
$\downarrow$

Even cycle / Acyclic

- Directed $\Rightarrow$ DAG
  - Need not be a directed tree.
- Undirected $\Rightarrow$ Tree
  - Can be coloured level by level

$\rightarrow$ Suppose we have BFS tree and layer computed for a given connected graph. Using this, can we infer bipartiteness?

ONLY Cross edge in the same layer will result in odd cycles.



{ s — u dist + s — v dist is even. }

{ So cross edge b/w u & v will result in odd cycle }

BFS(s).

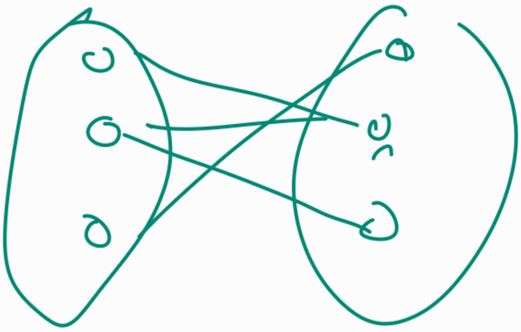
$L_0 \quad L_1 \quad L_2 \quad \dots \quad L_k$.

Part 1: $L_0 \cup L_2 \cup L_4 \cdots \cup L_k$

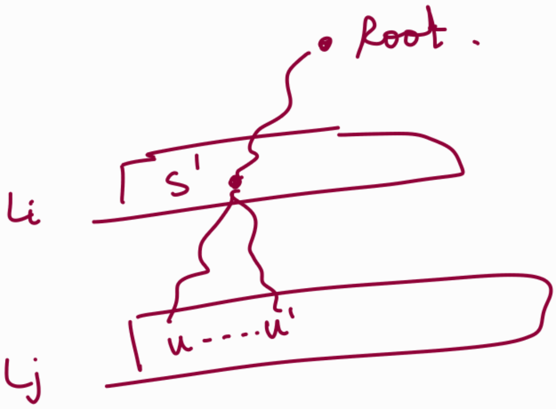
Part 2: $L_1 \cup L_3 \cup \cdots \cup L_{k-1}$

Never have edge b/w L_{k-2} & L_k.

Run BFS. Check cross edges. If no cross edges, then combine alternate layers to get two set.



If cross edge, then not bipartite.



Odd cycle.

$\because dist(s', u) = dist(s', u')$