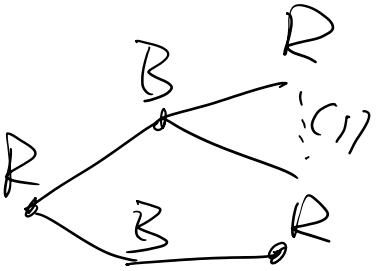
Thm We can test if a graph is bipartite in $O(m+n)$ time

Proof:

Since we can test bipartiteness separately for each connected component, we can assume the graph is connected



Algorithm:

- Run BFS starting at any vertex

(1) • If there is any edge in the same layer
  - output non-bipartite

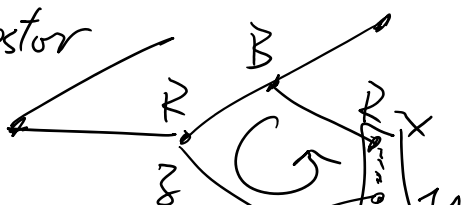
(2) • Else
  - output bipartite

Runtime: $O(m+n)$ Bottleneck: BFS.

Correctness:

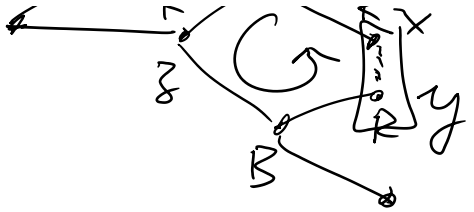
Case 1: There is edge $\{x,y\}$ on same layer

Let z be their common ancestor



$x \to z \to y \to x$

$x \to z \to y \to x$
is odd-length cycle



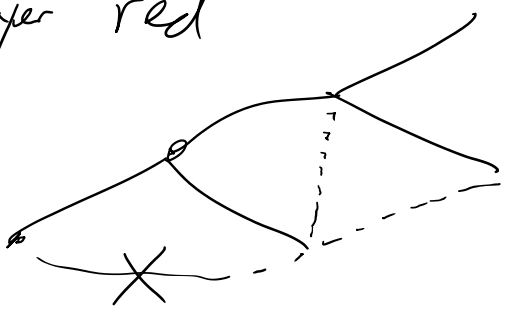
$\Rightarrow$ No way to 2-color the odd-length length

$\Rightarrow$ Original graph is non-bipartite.

Case 2: No edge on same layer.

Color nodes on even layer blue
odd layer red

Since no edge on same layer
and all edge join nodes
on same or adjacent layer [BFS properties]



The 2 coloring we give is valid.

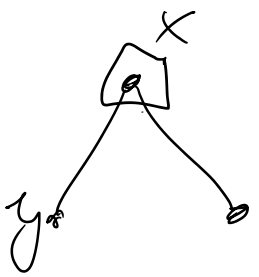
Corollary: A graph is non-bipartite
iff it has odd-length cycle.

Lemma For any edge $\{x, y\}$, for any DFS tree T

x is a ancestor of y or vice versa
(including child)

(including child)

Proof WLOG, x is discovered first.



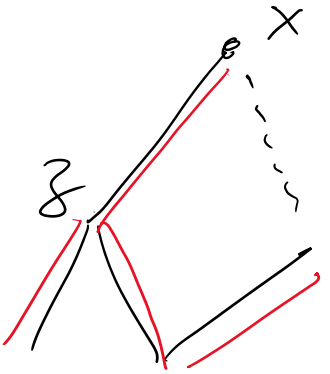
(without loss of generality)

Goal: x is ancestor of y.

We call DFS(x) at the true discovery x.

Case 1° y is child of x

Case 2° y is examined in the subcall.



$\Rightarrow$ x is a "true" ancestor of y