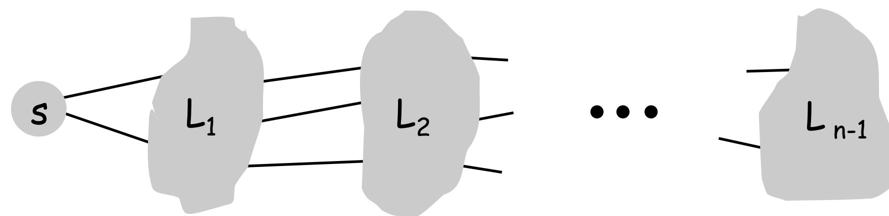
Breadth First Search

BFS intuition. Explore outward from s in all possible directions, adding nodes one "layer" at a time.



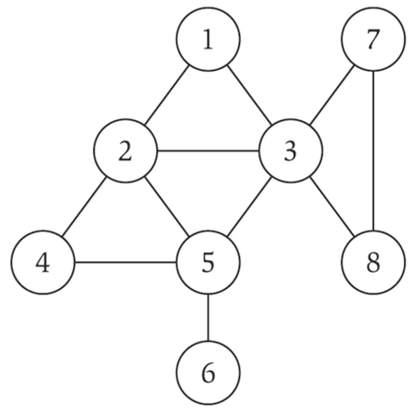
BFS algorithm.

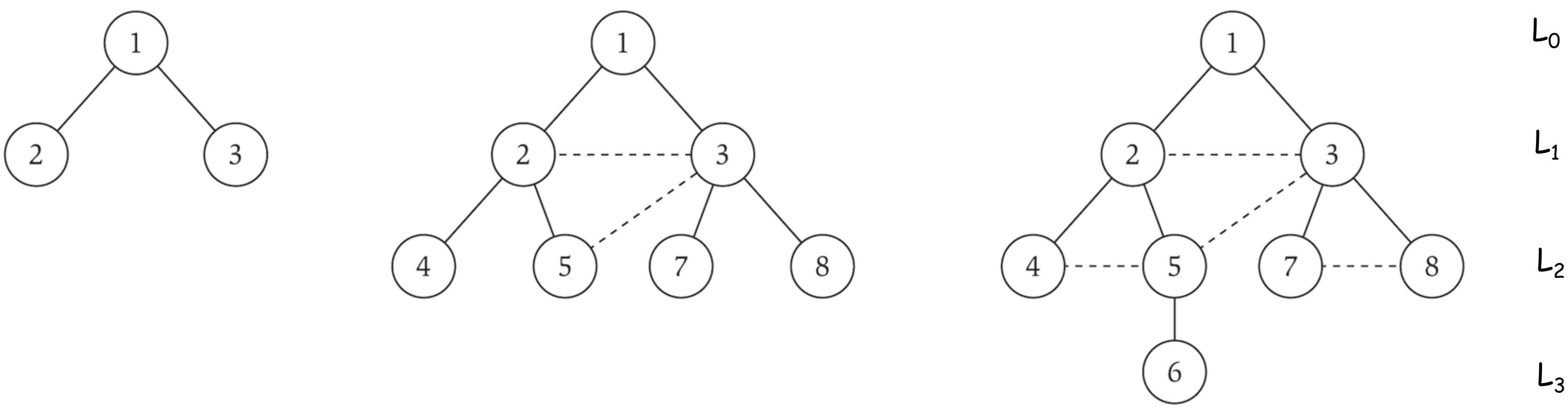
- $L_0 = \{ s \}$.
- L_1 = all neighbors of L_0.
- L_2 = all nodes that do not belong to L_0 or L_1, and that have an edge to a node in L_1.
- L_{i+1} = all nodes that do not belong to an earlier layer, and that have an edge to a node in L_i.

Theorem. For each i, L_i consists of all nodes at distance exactly i from s. There is a path from s to t iff t appears in some layer.

Breadth First Search

Property. Let T be a BFS tree of G = (V, E), and let (x, y) be an edge of G. Then the level of x and y differ by at most 1.





(a) (b) (c)

Breadth First Search: Analysis

Theorem. The above implementation of BFS runs in $O(m + n)$ time if the graph is given by its adjacency list representation.

Pf.

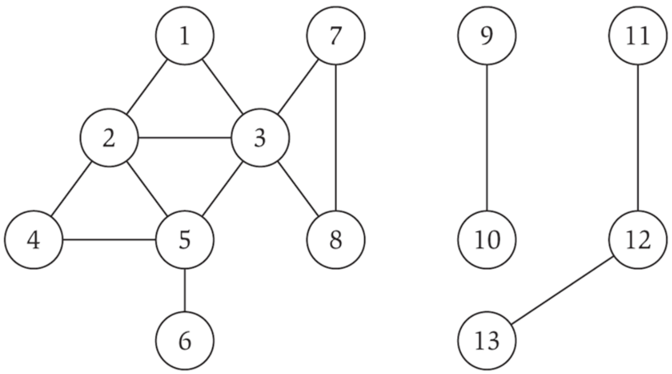
- Easy to prove $O(n^2)$ running time:
  - at most n lists L[i]
  - each node occurs on at most one list; for loop executed $\leq$ n times
  - when we consider node u, there are $\leq$ n incident edges (u, v), and we spend $O(1)$ processing each edge

- Actually runs in $O(m + n)$ time:
  - when we consider node u, there are deg(u) incident edges (u, v)
  - total time processing edges is $\Sigma_{u \in V} \deg(u) = 2m$ ▪

each edge (u, v) is counted exactly twice in sum: once in deg(u) and once in deg(v)

Connected Component

Connected component. Find all nodes reachable from s.



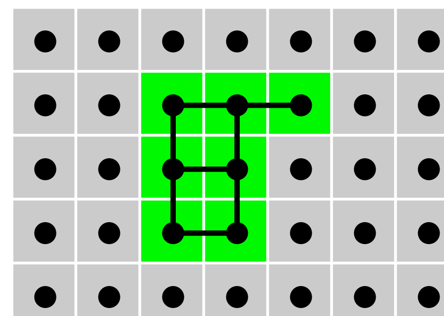
Connected component containing node 1 = { 1, 2, 3, 4, 5, 6, 7, 8 }.

Flood Fill

Flood fill. Given lime green pixel in an image, change color of entire blob of neighboring lime pixels to blue.

- Node: pixel.
- Edge: two neighboring lime pixels.
- Blob: connected component of lime pixels.

recolor lime green blob to blue

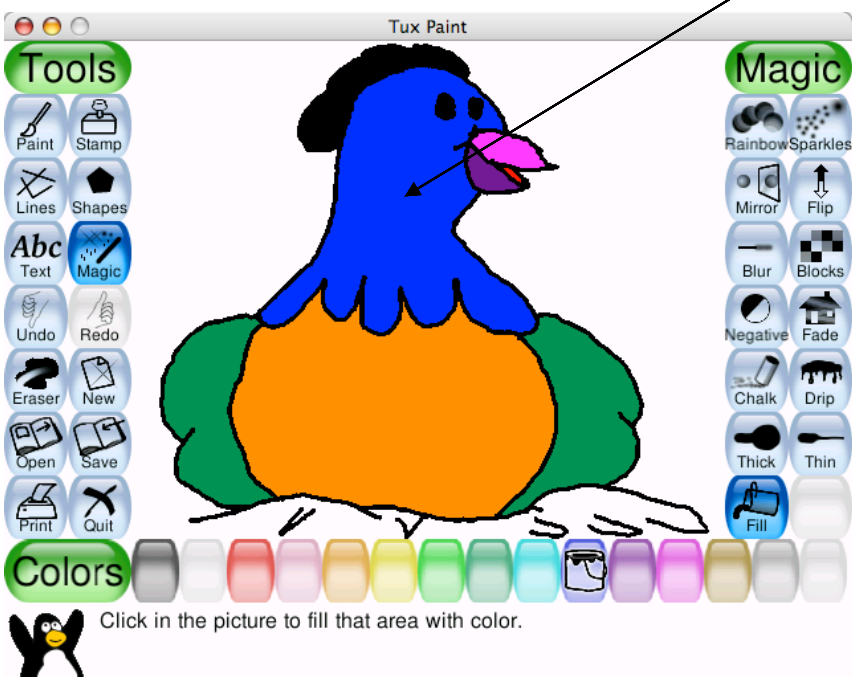


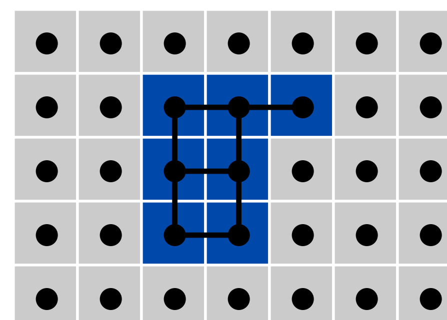
Flood Fill

Flood fill. Given lime green pixel in an image, change color of entire blob of neighboring lime pixels to blue.

- Node: pixel.
- Edge: two neighboring lime pixels.
- Blob: connected component of lime pixels.

recolor lime green blob to blue

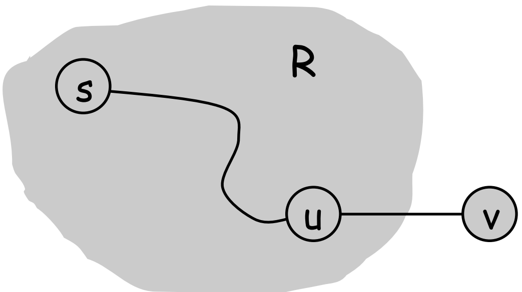




Connected Component

Connected component. Find all nodes reachable from s.

R will consist of nodes to which s has a path
Initially $R = \{s\}$
While there is an edge (u, v) where $u \in R$ and $v \notin R$
  Add v to R
Endwhile



it's safe to add v

Theorem. Upon termination, R is the connected component containing s.
- BFS = explore in order of distance from s.
- DFS = explore in a different way.

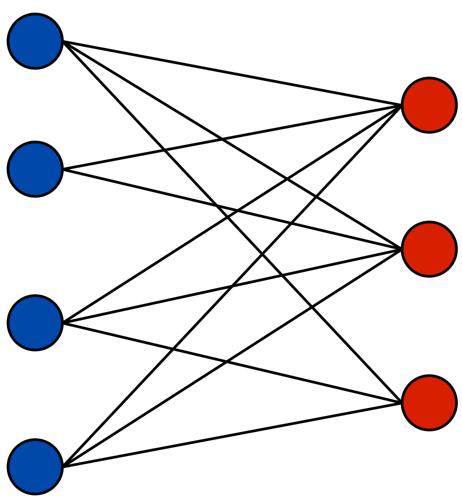
3.4 Testing Bipartiteness

Bipartite Graphs

Def. An undirected graph G = (V, E) is bipartite if the nodes can be colored red or blue such that every edge has one red and one blue end.

Applications.

- Stable marriage: men = red, women = blue.
- Scheduling: machines = red, jobs = blue.

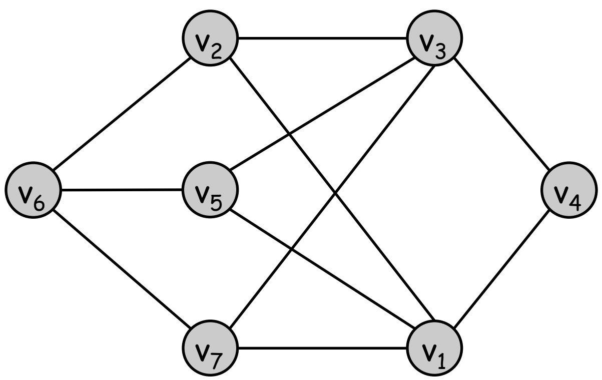


a bipartite graph

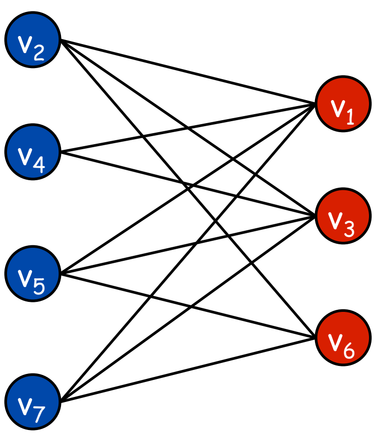
Testing Bipartiteness

Testing bipartiteness. Given a graph G, is it bipartite?

- Many graph problems become:
  - easier if the underlying graph is bipartite (matching)
  - tractable if the underlying graph is bipartite (independent set)
- Before attempting to design an algorithm, we need to understand structure of bipartite graphs.



a bipartite graph G

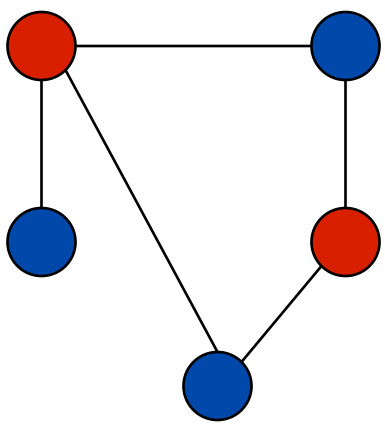


another drawing of G

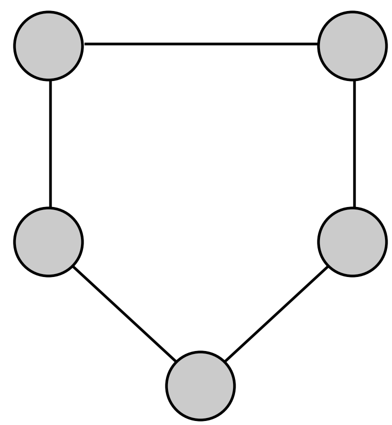
An Obstruction to Bipartiteness

Lemma. If a graph G is bipartite, it cannot contain an odd length cycle.

Pf. Not possible to 2-color the odd cycle, let alone G.



bipartite
(2-colorable)



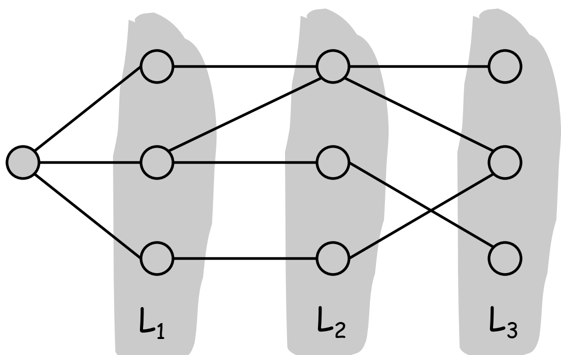
not bipartite
(not 2-colorable)

Bipartite Graphs

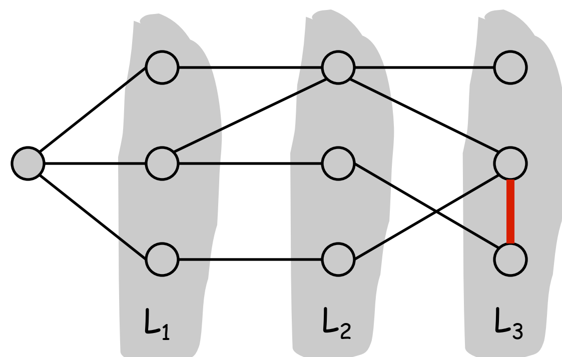
Lemma. Let G be a connected graph, and let $L_0, \ldots, L_k$ be the layers produced by BFS starting at node s. Exactly one of the following holds.

(i) No edge of G joins two nodes of the same layer, and G is bipartite.

(ii) An edge of G joins two nodes of the same layer, and G contains an odd-length cycle (and hence is not bipartite).



Case (i)



Case (ii)

Bipartite Graphs

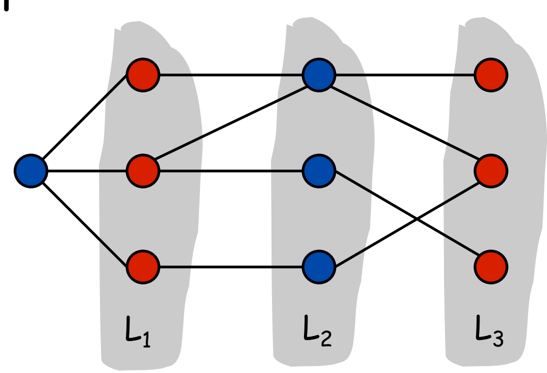
Lemma. Let G be a connected graph, and let $L_0, \ldots, L_k$ be the layers produced by BFS starting at node s. Exactly one of the following holds.

(i) No edge of G joins two nodes of the same layer, and G is bipartite.

(ii) An edge of G joins two nodes of the same layer, and G contains an odd-length cycle (and hence is not bipartite).

Pf. (i)

- Suppose no edge joins two nodes in the same layer.
- By previous lemma, this implies all edges join nodes on same level.
- Bipartition: red = nodes on odd levels, blue = nodes on even levels.



Case (i)

Bipartite Graphs

Lemma. Let G be a connected graph, and let $L_0, \ldots, L_k$ be the layers produced by BFS starting at node s. Exactly one of the following holds.

(i) No edge of G joins two nodes of the same layer, and G is bipartite.

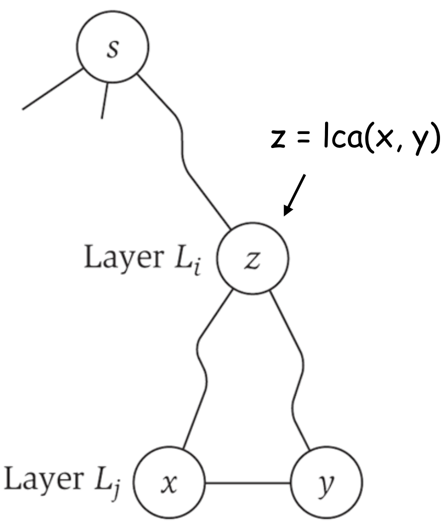
(ii) An edge of G joins two nodes of the same layer, and G contains an odd-length cycle (and hence is not bipartite).

Pf. (ii)

- Suppose (x, y) is an edge with x, y in same level L_j.
- Let z = lca(x, y) = lowest common ancestor.
- Let L_i be level containing z.
- Consider cycle that takes edge from x to y, then path from y to z, then path from z to x.

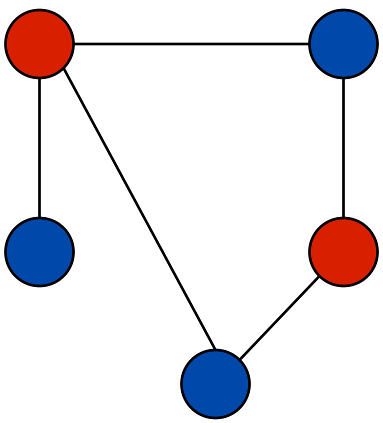
(x, y) | path from y to z | path from z to x

- Its length is 1 + (j-i) + (j-i), which is odd. ▪

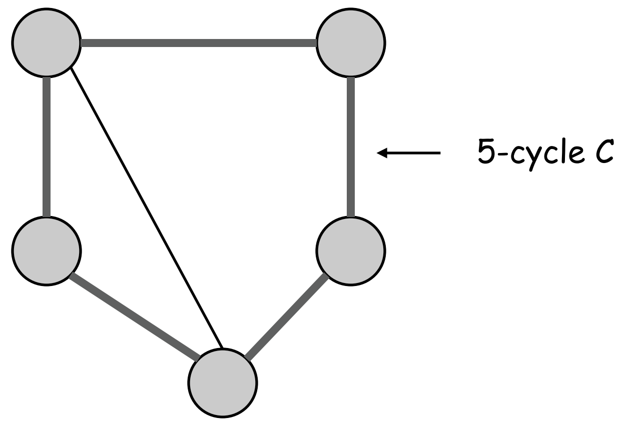


Obstruction to Bipartiteness

Corollary. A graph G is bipartite iff it contain no odd length cycle.



bipartite
(2-colorable)



not bipartite
(not 2-colorable)