Testing Bipartiteness: An Application of Breadth-First Search

Section 3.4

Bipartite Graphs

- **Graph G = (V,E) is bipartite if**
  - **V can be partitioned into sets X and Y in such a way that every edge has one end-point in X and other end-point in Y**

Bipartite Graphs

- Graph G = (V,E) is bipartite if
  - V can be partitioned into sets X and Y in such a way that every edge has one end-point in X and other end-point in Y

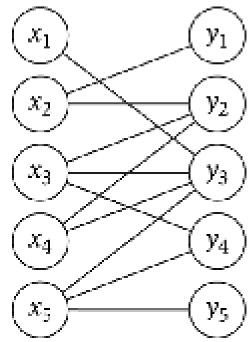


Figure 1.5 A bipartite graph.

Bipartite Graphs

- Graph G = (V,E) is bipartite if
  - V can be partitioned into sets X and Y in such a way that every edge has one end-point in X and other end-point in Y

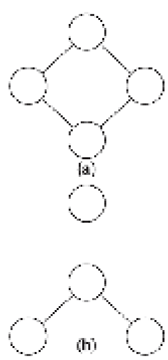


Figure 1.3 Each of (a) and (b) depicts a graph on four nodes.

Bipartite Graphs: Examples

- X = set of students
- Y = set of classes
- Edge (u,v): if and only if student u is enrolled in class v

Bipartite Graphs: Examples

- X = set of TA applicants
- Y = set of classes
- Edge (u,v): if and only if applicant u is eligible for teaching class v

Bipartite Graphs: Examples

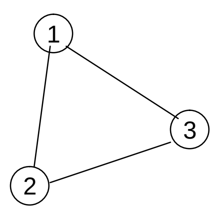
- X = set of TA applicants
- Y = set of classes
- Edge (u,v): if and only if applicant u is eligible for teaching class v
- Our graduate secretary handles this graph every semester

Bipartite Graphs

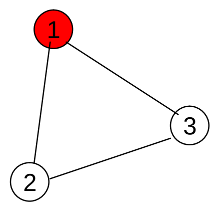
- X = set of red nodes
- Y = set of blue nodes
- Edges are between red and blue nodes
  - No edge between red nodes
  - No edge between blue nodes

What graphs are not bipartite?

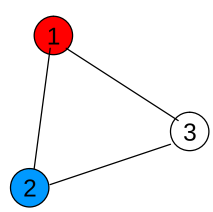
What graphs are not bipartite?



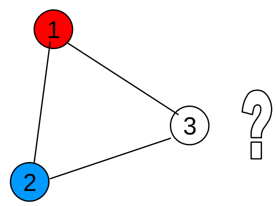
What graphs are not bipartite?



What graphs are not bipartite?



What graphs are not bipartite?



What graphs are not bipartite?

- More generally, odd cycles have problems
  - Consider cycle with nodes 1, 2, … 2k + 1
  - Node 1 = red
  - Node 2 = blue
  - Node 3 = red
  - …
  - Node 2k = blue
  - Node 2k + 1 = ?

What graphs are not bipartite?

- **[3.14] If a graph G is bipartite, then it cannot contain an odd cycle**

What graphs are not bipartite?

- **[3.14] If a graph G is bipartite, then it cannot contain an odd cycle**

  - **But is odd cycle the only obstacle?**

What graphs are not bipartite?

- **[3.14] If a graph G is bipartite, then it cannot contain an odd cycle**
  - **But is odd cycle the only obstacle?**
  - **If no odd cycle, is there a valid red/blue coloring of the nodes?**
    - **i.e., every edge has a red and blue end-point.**

What graphs are not bipartite?

- **[3.14] If a graph G is bipartite, then it cannot contain an odd cycle**
  - **But is odd cycle the only obstacle?**
  - **If no odd cycle, is there a valid red/blue coloring of the nodes?**
    - **i.e., every edge has a red and blue end-point.**

How can we know if a given graph is bipartite?

An algorithm for testing "bipartiteness"

- Choose a vertex s and color it red
- Color neighbors of s blue
- Color neighbors of *these* nodes red
- … and so on until all nodes are colored

An algorithm for testing "bipartiteness"

- Choose a vertex s and color it red
- Color neighbors of s blue
- Color neighbors of *these* nodes red
- … and so on until all nodes are colored

- Do we have a valid coloring?
  - Yes: bipartite
  - No: not bipartite

BFS Coloring

- **Start exploring from *s***
- **Color *s* red**
- **Color layer L_1 blue, L_2 red, and so on…**
- **Color odd layers blue and even layers red**
- **Easy to implement on top of BFS!**

BFS Coloring

```
BFS(s):
  Set Discovered[s] = true and Discovered[v] = false for all other v
  Initialize L[0] to consist of the single element s
  Set the layer counter i = 0
  Set the current BFS tree T = ∅
  While L[i] is not empty
    Initialize an empty list L[i + 1]
    For each node u ∈ L[i]
      Consider each edge (u, v) incident to u
      If Discovered[v] = false then
        Set Discovered[v] = true
        Add edge (u, v) to the tree T
        Add v to the list L[i + 1]
      Endif
    Endfor
    Increment the layer counter i by one
  Endwhile
```

BFS Coloring

```
BFS(s):
  Set Discovered[s] = true and Discovered[v] = false for all other v
  Initialize L[0] to consist of the single element s
  Set the layer counter i = 0
  Set the current BFS tree T = ∅
  While L[i] is not empty
    Initialize an empty list L[i+1]
    For each node u ∈ L[i]
      Consider each edge (u,v) incident to u
      If Discovered[v] = false then
        Set Discovered[v] = true
        Add edge (u,v) to the tree T
        Add v to the list L[i + 1]
      Endif
    Endfor
    Increment the layer counter i by one
  Endwhile
```

Set color[v] here

Analysis of BFS coloring

- [3.15] Let *G* be a connected graph, and let L_1, L_2, … be the layers produced by BFS starting at node s. Then exactly one of the following two things hold.
  1. There is no edge of *G* joining two nodes of the same layer. In this case *G* is a bipartite graph in which the nodes in even-numbered layers can be colored red, and the nodes in odd-numbered layers can be colored blue
  2. There is an edge of *G* joining two nodes of the same layer. In this case, *G* contains an odd-cycle, and so it cannot be bipartite.