Bipartite Graphs

Outline

This topic looks at another problem solved by breadth-first traversals

- Determining if a graph is bipartite
- Definition of a bipartite graph
- The algorithm
- An example

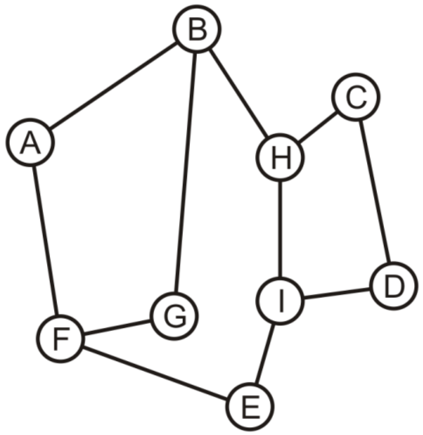
Definition

Definition

– A *bipartite graph* is a graph where the vertices V can be divided into two disjoint sets V_1 and V_2 such that **every** edge has one vertex in V_1 and the other in V_2

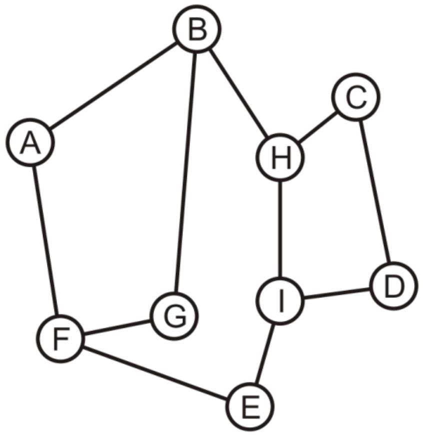
Bipartite Graphs

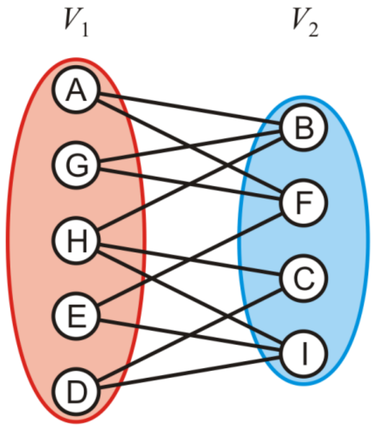
Consider this graph: is it bipartite?



Bipartite Graphs

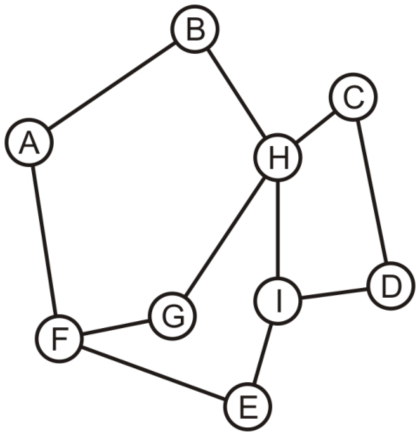
Yes: With a little work, it is possible to determine that we can decompose the vertices into two disjoint sets





Bipartite Graphs

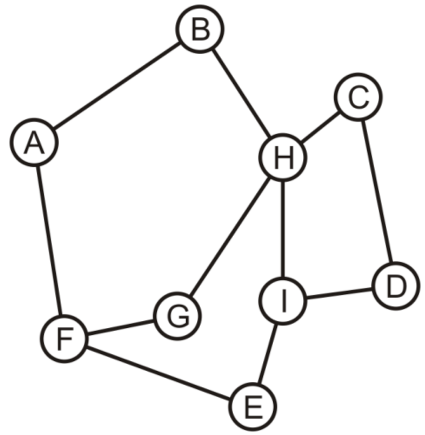
Is this graph bipartite?



Bipartite Graphs

In this case, it is not a bipartite graph

– Can we find a traversal that will determine if a graph is bipartite?



Bipartite Graphs

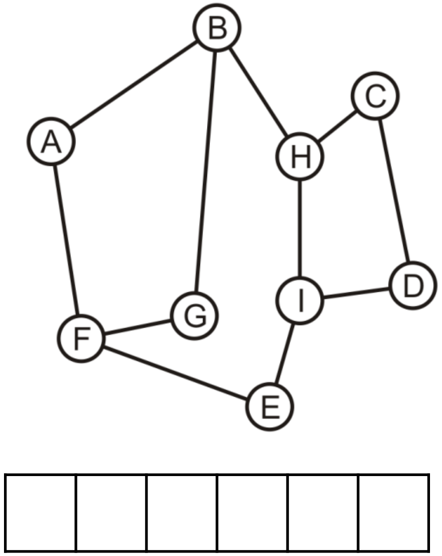
Consider using a breadth-first traversal for a connected graph:

- Choose a vertex, mark it belonging to V_1 and push it onto a queue
- While the queue is not empty, pop the front vertex v and
  - Any adjacent vertices that are already marked must belong to the set not containing v, otherwise, the graph is not bipartite (we are done); while
  - Any unmarked adjacent vertices are marked as belonging to the other set and they are pushed onto the queue
- If the queue is empty, the graph is bipartite

Bipartite Graphs

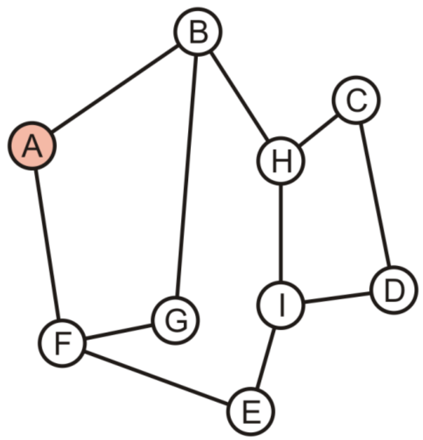
With the first graph, we can start with any vertex

- We will use colours to distinguish the two sets



Bipartite Graphs

Push A onto the queue and colour it red

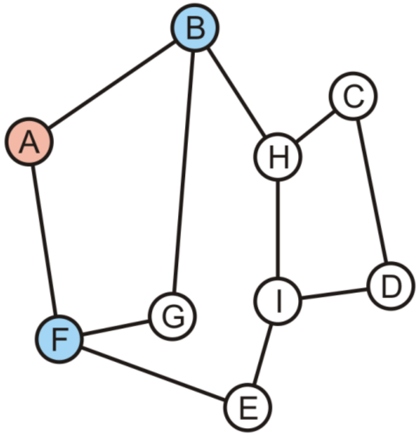


A					

Bipartite Graphs

Pop A and its two neighbours are not marked:

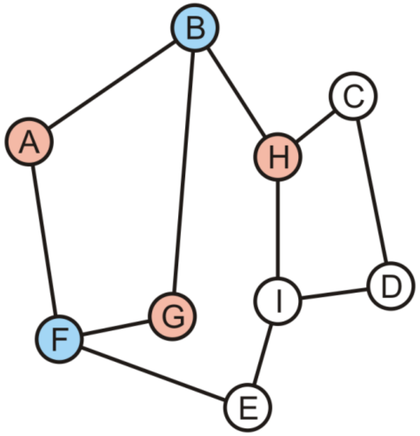
– Mark them as blue and push them onto the queue



B	F				

Bipartite Graphs

Pop B—it is blue:
- Its one marked neighbour, A, is red
- Its other neighbours G and H are not marked: mark them red and push them onto the queue

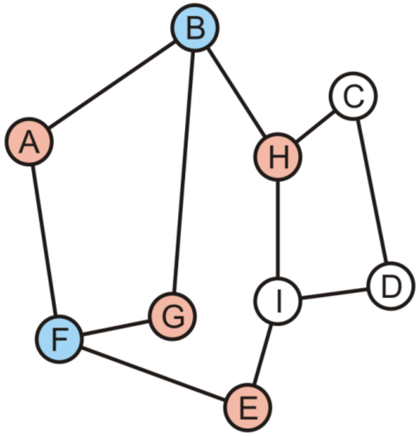


F	G	H			

Bipartite Graphs

Pop F—it is blue:

- Its two marked neighbours, A and G, are red
- Its neighbour E is not marked: mark it red and pus it onto the queue

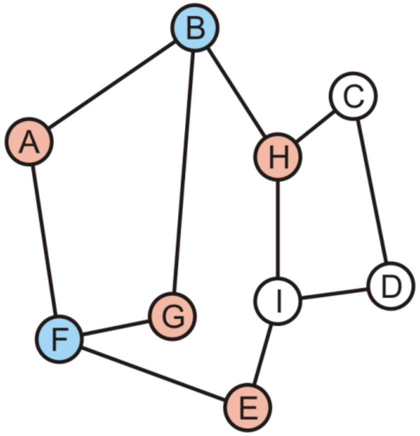


G	H	E			

Bipartite Graphs

Pop G—it is red:

– Its two marked neighbours, B and F, are blue

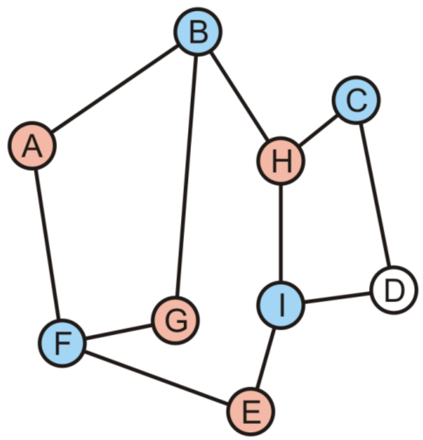


H	E				

Bipartite Graphs

Pop H—it is red:

- Its marked neighbours, B, is blue
- It has two unmarked neighbours, C and I; mark them blue and push them onto the queue

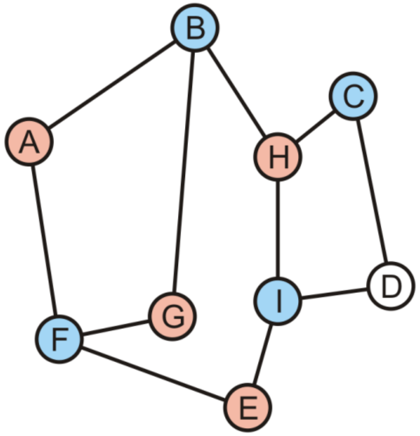


E	C	I			

Bipartite Graphs

Pop E—it is red:

- Its marked neighbours, F and I, are blue

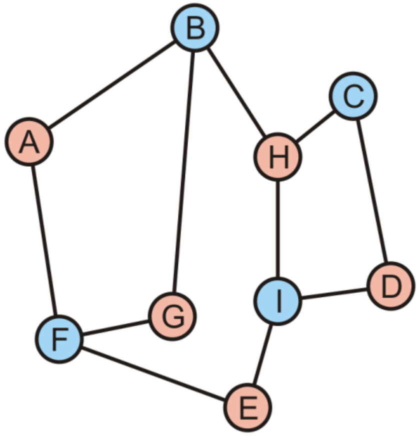


C	I				

Bipartite Graphs

Pop C—it is blue:

- Its marked neighbour, H, is red
- Mark D as red and push it onto the queue

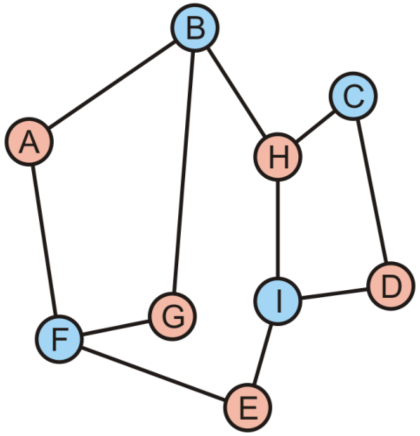


I	D				

Bipartite Graphs

Pop I—it is blue:

– Its marked neighbours, H, D and E, are all red

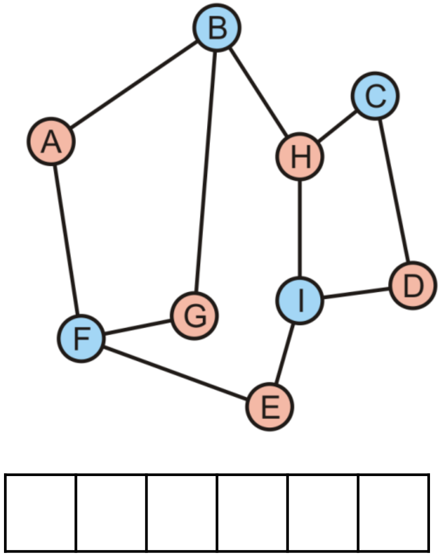


D					

Bipartite Graphs

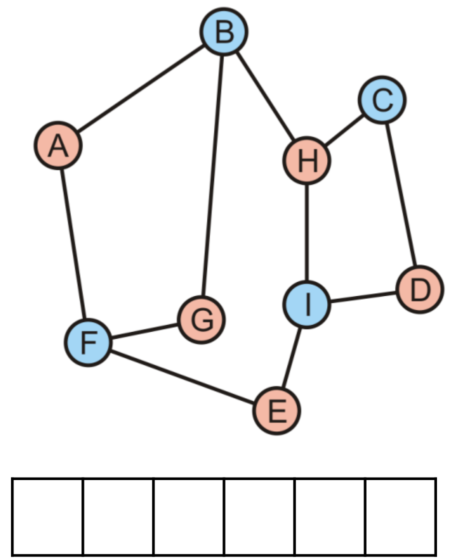
Pop D—it is red:

- Its marked neighbours, C and I, are both blue



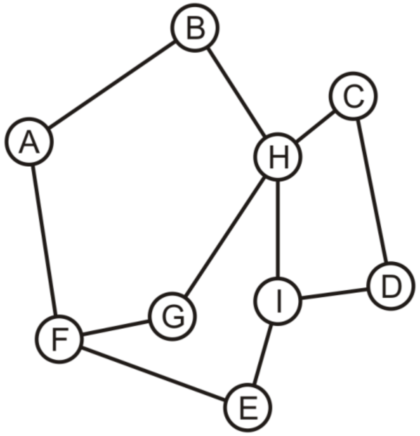
Bipartite Graphs

The queue is empty, the graph is bipartite



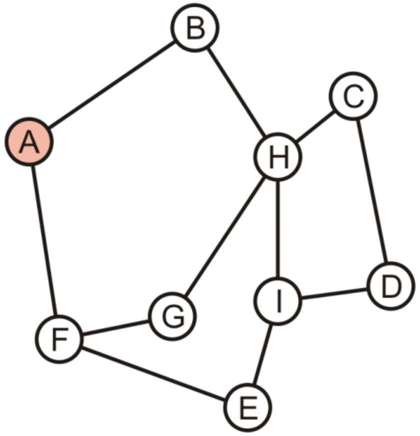
Bipartite Graphs

Consider the other graph which was claimed to be not bipartite



Bipartite Graphs

Push A onto the queue and colour it red

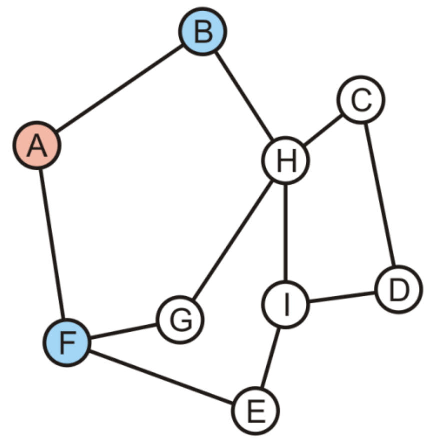


A					

Bipartite Graphs

Pop A off the queue:

- Its neighbours are unmarked: colour them blue and push them onto the queue

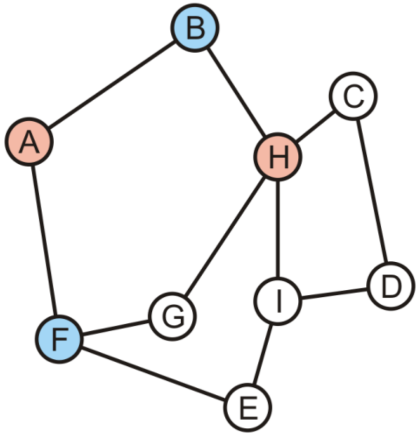


B	F				

Bipartite Graphs

Pop B off the queue:

- Its one neighbour, A, is red
- The other neighbour, H, is unmarked: colour it red and push it onto the queue

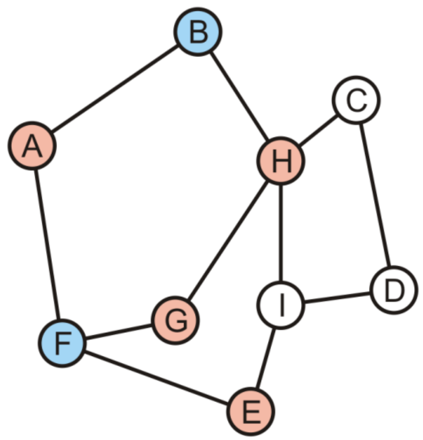


F	H				

Bipartite Graphs

Pop F off the queue:

- Its one neighbour, A, is red
- The other neighbours, E and G, are unmarked: colour them red and push it onto the queue

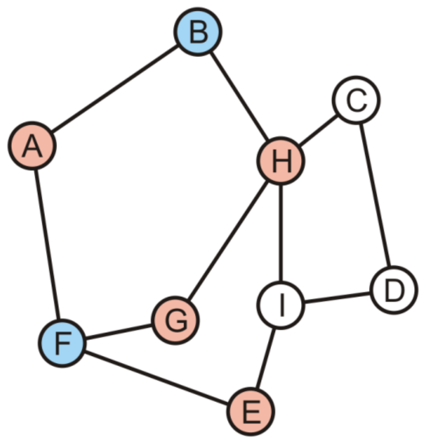


H	E	G			

Bipartite Graphs

Pop H off the queue—it is red:

- Its one neighbour, G, is already red
- The graph is not bipartite



E	G				

Bipartite Graphs

Definition

Cycles that contains either an even number or an odd number of vertices are said to be *even cycles* and *odd cycles*, respectively

Theorem

A graph is bipartite if and only if it does not contain any odd cycles

Reference: Kleinberg and Tardos

Sumary

This topic looked at identifying bipartite graphs

- Perform a breadth-first traversal
- Each vertex is given one of two identifiers (we used color)
- The first vertex is identified as one color and pushed onto the queue
- When a vertex is popped:
    - Each unvisited neighbor is pushed onto the tree with the opposite color
    - Each visited neighbor must be the opposite color
        - If one is not, the graph is not bipartite

References

Wikipedia, http://en.wikipedia.org/wiki/Breadth-first_search#Testing_bipartiteness
http://en.wikipedia.org/wiki/Breadth-first_search
http://en.wikipedia.org/wiki/Bipartite_graph

[1] Jon Kleinberg and Éva Tardos, *Algorithm Design,* Addison Wesley, 2006, §§3.2-5, pp.78-99.