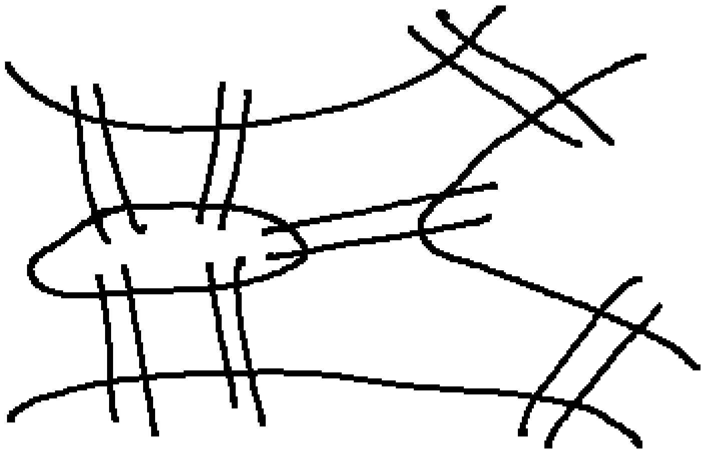
Further Data Structures

- The story so far
  - Saw some fundamental operations as well as advanced operations on arrays, stacks, and queues
  - Saw a dynamic data structure, the linked list, and its applications.
  - Saw the hash table so that insert/delete/find can be supported efficiently.
  - Saw trees and and applications to searching.
- This week we will
  - Introduce graphs as a data structure.
  - Study operations on graphs including searching.

Introduction to Graphs

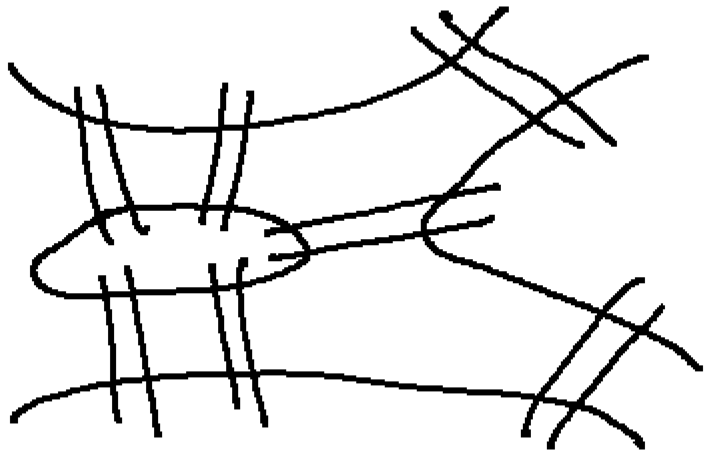
- Consider the following problem.
- A river with an island and bridges.
- The problem is to see if there is a way to start from some landmass and using each bridge exactly once, return to the starting point.

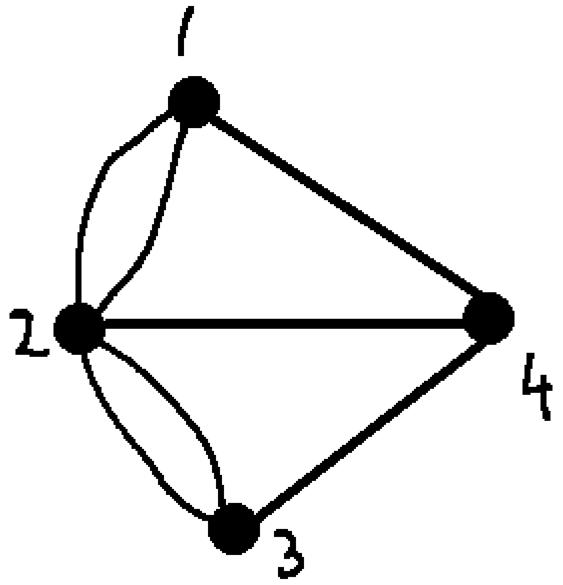


Introduction to Graphs

- The above problem dates back to the 17th century.
- Several people used to try to solve it.
- Euler showed that no solution exists for this problem.
- Further, he exactly characterized when a solution exists.
- By solving this problem, it is said that Euler started the study of graphs.

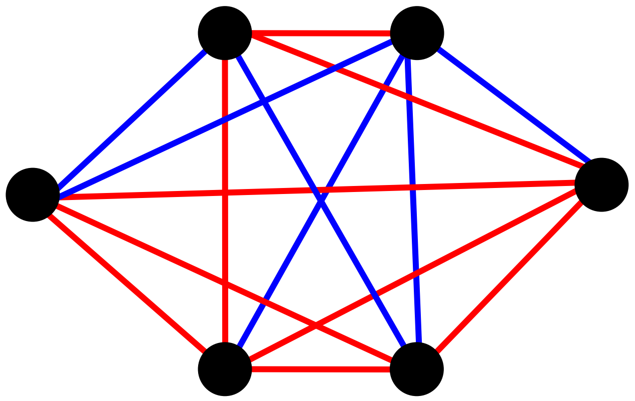
Introduction to Graphs





- The figure on the right shows the same situation modeled as a graph.
- There exist several such classical problems where graph theory has been used to arrive at elegant solutions.

Introduction to Graphs



- Another such problem: In any set of at least six persons, there are either three mutual acquaintances or three mutual strangers.

Introduction to Graphs

- Formally, let V be a set of points, also called as vertices.
- Let $E \subseteq V x V$ be a subset of the cross product of V with itself. Elements of E are also called as edges.
- A graph can be seen as the tuple (V, E). Usually denoted by upper case letters G, H, etc.

Our Interest

- Understand a few terms associated with graphs.
- Study how to represent graphs in a computer program.
- Study how traverse graphs.
- Study mechanisms to find paths between vertices.
- Spanning trees
- And so on...

Few Terms

- Recall that a graph G = (V, E) is a tuple with E being a subset of VxV.
- Scope for several variations: for u, v in V
  - Should we treat (u,v) as same as (v,u)?

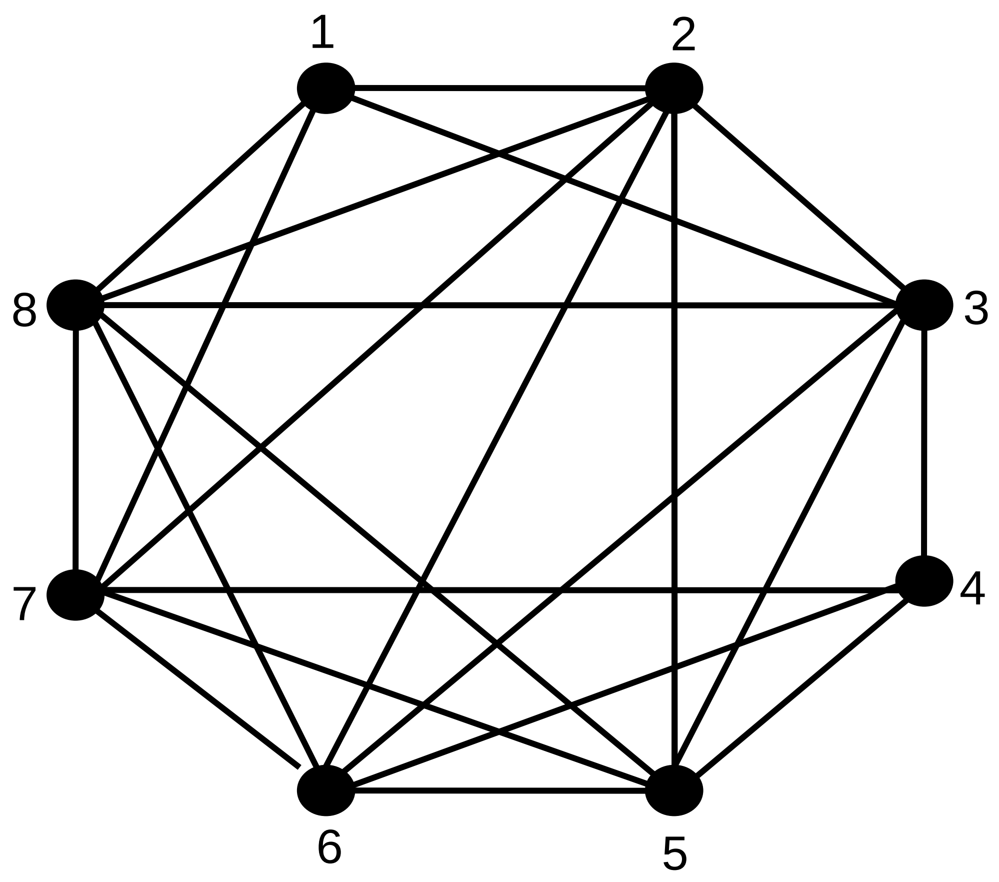
Few Terms

- Recall that a graph G = (V, E) is a tuple with E being a subset of VxV.
- Scope for several variations: for u, v in V
  - Should we treat (u,v) as same as (v,u)? In this case, the graph is called as a undirected graph.
  - Treat (u,v) as different from (v,u).

Few Terms

- Recall that a graph G = (V, E) is a tuple with E being a subset of VxV.
- Scope for several variations: for u, v in V
  - Should we treat (u,v) as same as (v,u)? In this case, the graph is called as a undirected graph.
  - Treat (u,v) as different from (v,u). In this case, the graph is called as a directed graph.
  - Should we allow (u,u) in E? Edges of this kind are called as self-loops.

Undirected Graphs



- In this case, the edge (u,v) is same as the edge (v,u).
  - Normally written as edge uv.

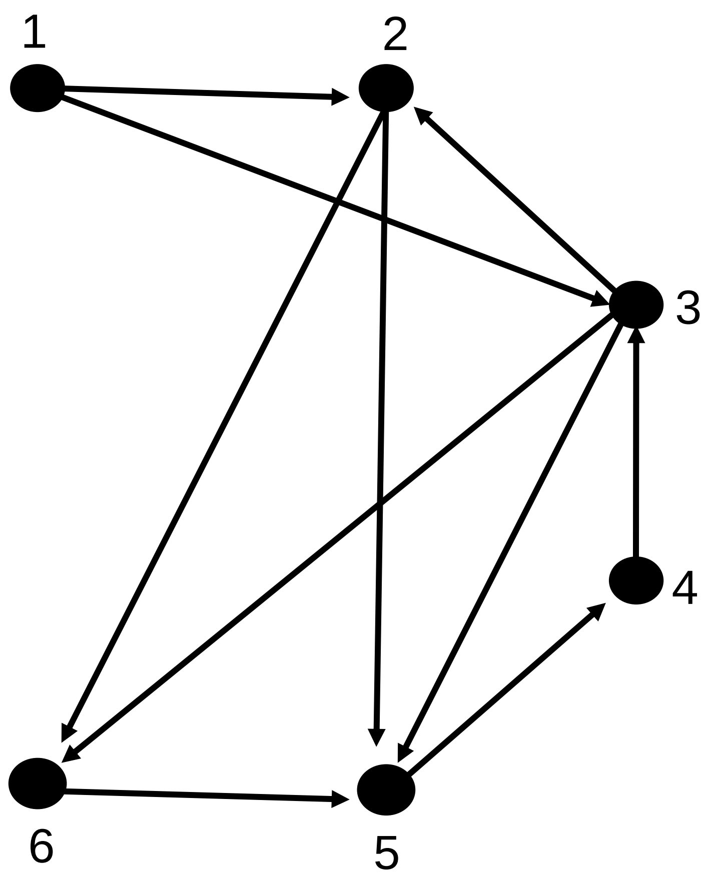
Undirected Graphs

- The degree of a node v in a graph G = (V,E) is the number of its neighbours.
  - It is denoted by d(v).
- In the above example, the degree of vertex 4 is 4. The neighbors of vertex 4 are {3, 5, 6, 7}.
- The degree of a graph G = (V,E) is the maximum degree of any node in the graph and is denoted $\Delta(G)$. Sometimes, written as just Δ when G is clear from the context.
  - Thus, $\Delta = \max_{v \in V} d(v)$.
  - Thus $\Delta = 6$ for the above graph.

Some Terms

- In a graph $G = (V,E)$, a path is a sequence of vertices $v_1, v_2, \cdot \cdot \cdot , v_i$, all distinct, such that $v_k v_{k+1} \in E$ for $1 \leq k \leq i - 1$.
- If, under the above conditions, $v_1 = v_i$ then it is called a cycle.
- The length of such a path(cycle) is $i - 1$(resp. i).
- An example: 3 – 8 – 5 – 2 in the above graph is a path from vertex 3 to vertex 2.
- Similarly, 2 – 7 – 6 – 5 – 2 is a cycle.

Directed Graphs



- In this case, the edge (u,v) is distinct from the edge (v,u).
  - Normally written as edge ⟨u, v⟩.

Directed Graphs

- Have to alter the definition of degree as
- in-degree(v) : the number of neighbors w of v such that (w,v) in E.
- out-degree(v) : the number of neighbors w of v such that (v,w) in E.
- in-degree(4) = 1
- out-degree(2) = 2.

Directed Graphs

- Have to alter the definition of path and cycle to directed path and directed cycle.

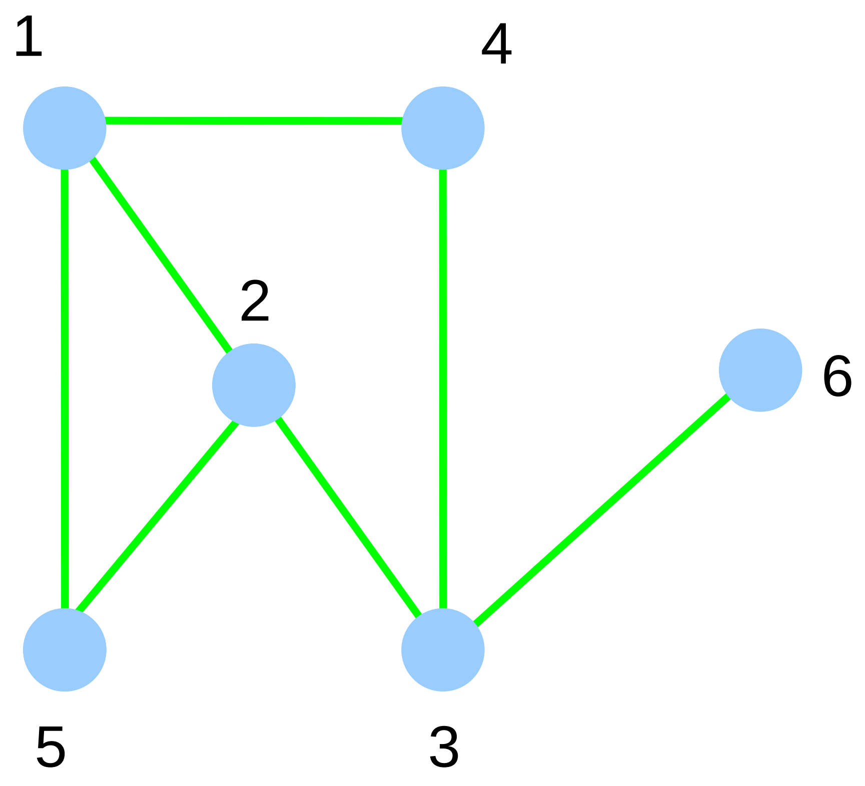
Representing Graphs

- How to represent graphs in a computer program.
- Several ways possible.

Adjacency Matrix

- The graph is represented by an $n \times n$–matrix where n is the number of vertices.
- Let the matrix be called A. Then the element A[i, j] is set to 1 if $(i, j) \in E(G)$ and 0 otherwise, where $1 \leq i, j \leq n$.
- The space required is $O(n^2)$ for a graph on n vertices.
- By far the simplest representation.
- Many algorithms work very efficiently under this representation.

Adjacency Matrix Example



A	1	2	3	4	5	6
1	0	1	0	1	1	0
2	1	0	1	0	1	0
3	0	1	0	1	0	1
4	1	0	1	0	0	0
5	1	1	0	0	0	0
6	0	0	1	0	0	0

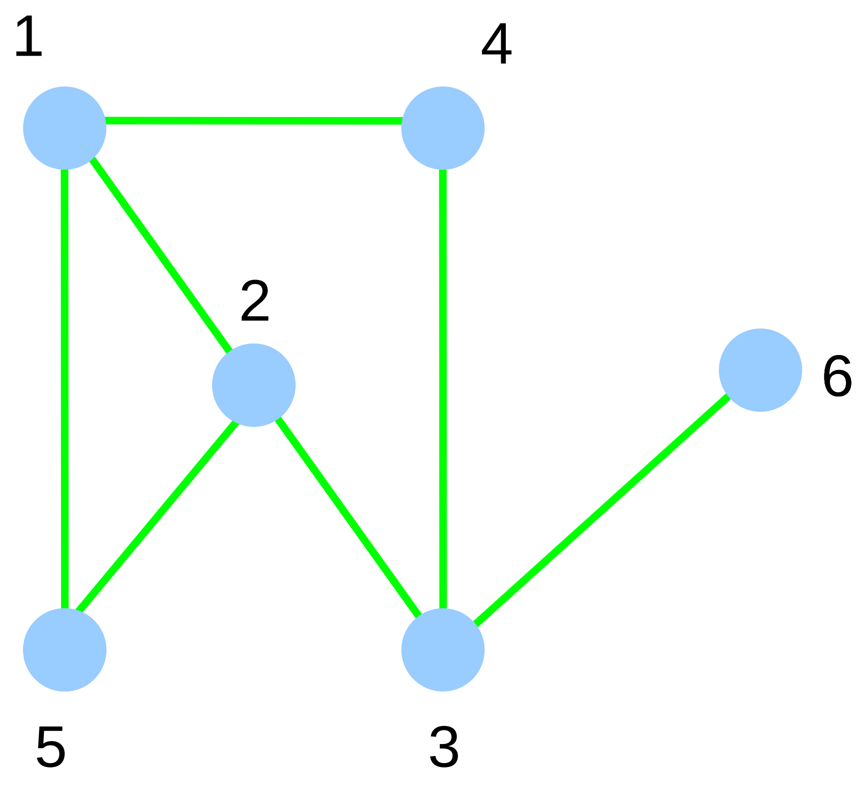
Adjacency Matrix Observations

- Space required is n^2
- The matrix is symmetric and 0,1—valued.
  - For directed graphs, the matrix need not be symmetric.
- Easy to check for any u,v whether uv is an edge.
- Most algorithms also take $O(n^2)$ time in this representation.
- The following is an exception: The Celebrity Problem.

Adjacency List

- Imagine a list for each vertex that will contain the list of neighbours of that vertex.
- The space required will only be O(m).
- However, one drawback is that it is difficult to check whether a particular pair (i, j) is an edge in the graph or not.

Adjacency List Example



$1 \rightarrow 2 \rightarrow 5 \rightarrow 4$

$2 \rightarrow 5 \rightarrow 1 \rightarrow 3$

$3 \rightarrow 2 \rightarrow 6 \rightarrow 4$

$4 \rightarrow 1 \rightarrow 3$

$5 \rightarrow 1 \rightarrow 2$

$6 \rightarrow 3$

Adjacency List

- Useful representation for sparse graphs.
- Extends to also directed graphs.

Other Representations

- Neighbor maps

Searching Graphs

- A fundamental problem in graphs. Also called as traversing a graph.
- Need to visit every vertex.
- Can understand several properties of a graph using a traversal.
- Two main techniques : breadth first search, and depth first search.

Breadth First Search

- Recall level order traversal of a tree.
  - Starting from the root, visits every vertex in a level by level manner.
- Let us develop breadth first search as an extension of level order traversal.
- A few questions to be answered before we develop breadth first search.

Breadth First Search

- Question 1: For a graph, no notion of a root vertex.
- So, where should BFS start from?

Breadth First Search

- Question 1: For a graph, no notion of a root vertex.
- So, where should BFS start from?
- So, have to specify a starting vertex. Typically denoted s.
- Still other problems exist.

Breadth First Search

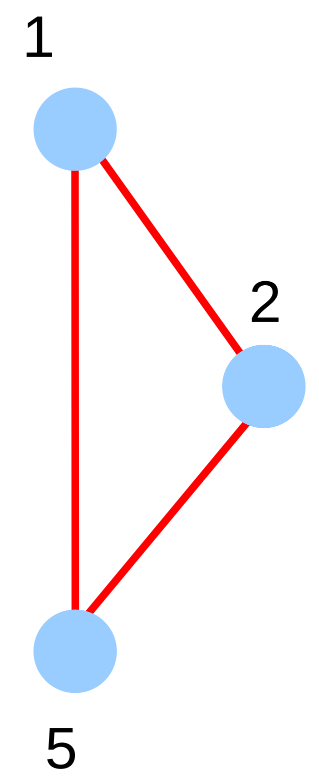
- In a tree, using level order traversal, each vertex is visited also exactly once.
  - Why?

Breadth First Search

- In a tree, using level order traversal, each vertex is visited also exactly once.
  - Recall that a tree is connected and has no cycles.

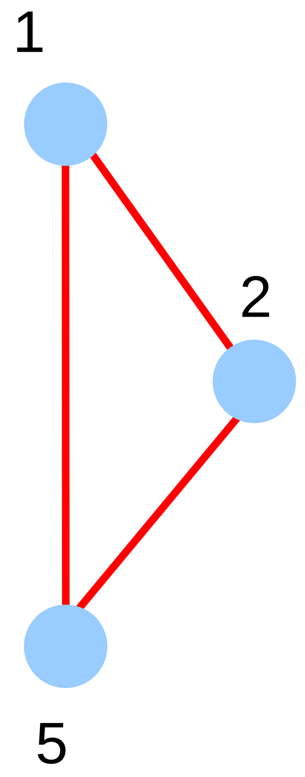
Breadth First Search

- In a tree, using level order traversal, each vertex is visited also exactly once.
  - Recall that a tree is connected and has no cycles.
- In a graph, that is no longer guaranteed.
  - Start from s = 2 and do a level order traversal.



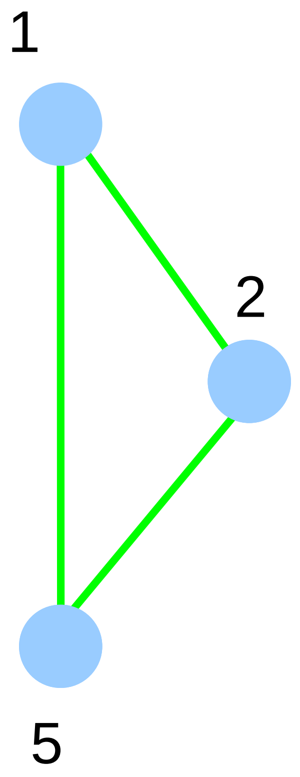
Breadth First Search

- In a tree, using level order traversal, each vertex is visited also exactly once.
  - Recall that a tree is connected and has no cycles.
- In a graph, that is no longer guaranteed.
  - Start from s = 2 and do a level order traversal
  - One of 1 or 5 visited more than once.



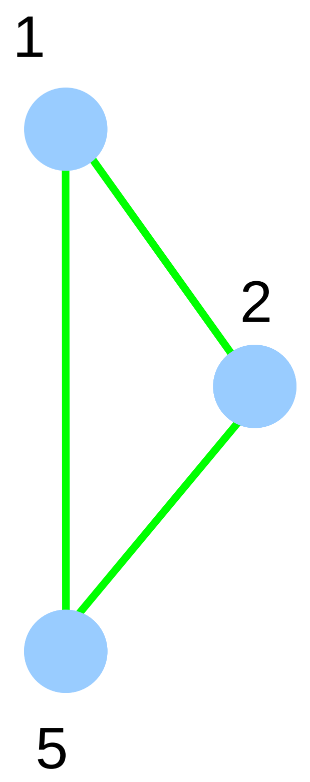
Breadth First Search

- Question 2: How to resolve that problem?



Breadth First Search

- Question 2: How to resolve that problem?
- Can remember if a vertex is already visited.
- Each vertex has a state among VISITED, NOT_VISITED, IN_PROGRESS.
- Why three states instead of just two?
  - Need them for a later use.

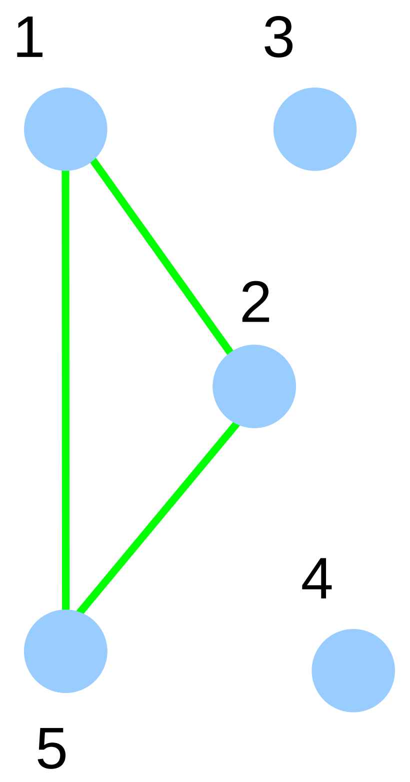


Breadth First Search

- Question 3: Can all vertices be reached from s?

Breadth First Search

- Question 3: Can all vertices be reached from s?
- For example, when s = 2, vertex 3 can never be visited.
- What to do with those vertices?
- Answer depends on the idea behind graph searching via BFS.



Breadth First Search

- The basic idea of breadth first search is to find the least number of edges between s and any other vertex in G.
  - The same property holds for level order traversal of a tree also with s as the root.
- Starting from s, we can thus visit vertices of distance k before visiting any vertex of distance k+1.
- For that purpose, define $d_s(v)$ to be the least number of edges between s and v in G.

Breadth First Search

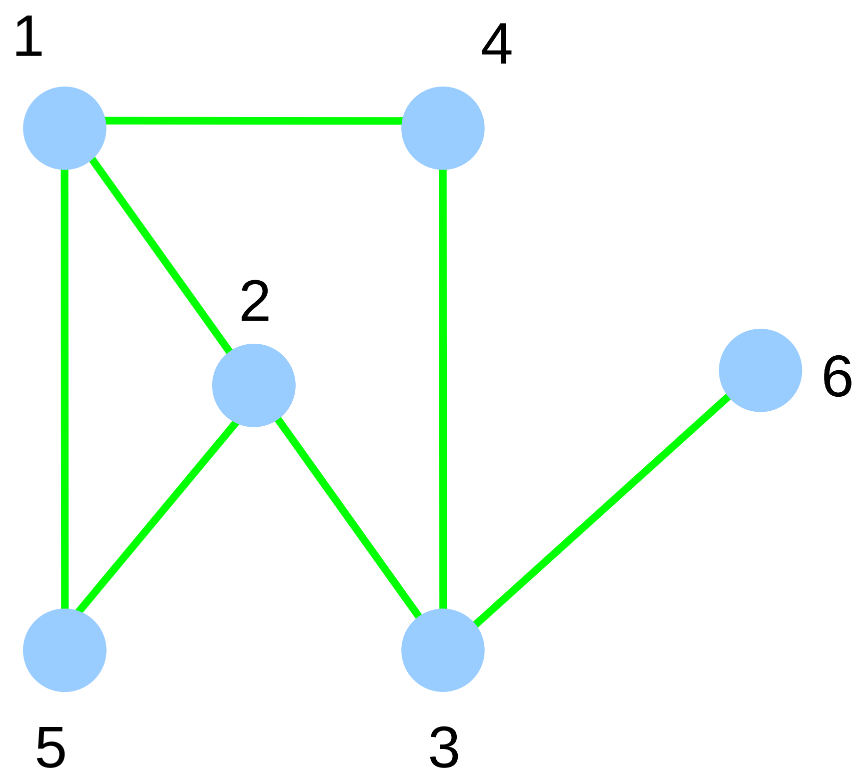
- So, for vertices v that are not reachable from s, can say that $d_s(v)$ is ▯.
- Alike a level order traversal of a tree, can use a queue to store vertices in progress.

BFS Procedure

```
Procedure BFS(G)
for each v ∈ V do
$\pi$(v)= NIL;state[v] = NOT_VISITED; d(v) = $\infty$;
End-for
d[s] = 0; state[s] = IN_PROGRESS; $\pi$[s]= NIL,
Q = EMPTY; Q.Enqueue(s);
While Q is not empty do
v = Q.Dequeue();
for each neighbour w of v do
    if state[w] = NOT_VISITED then
        state[w] = IN_PROGRESS; $\pi$[w] = v;
        d[w] = d[v] + 1; Q.Enqueue(w);
    end-if
end-for
state[v] = FINISHED
end-while
```

BFS Example

- Start from s = 2.



	1	2	3	4	5	6
d :	∞	0	∞	∞	∞	∞
π :	–	–	–	–	–	–

BFS – Additional Details

- What is the runtime of BFS?

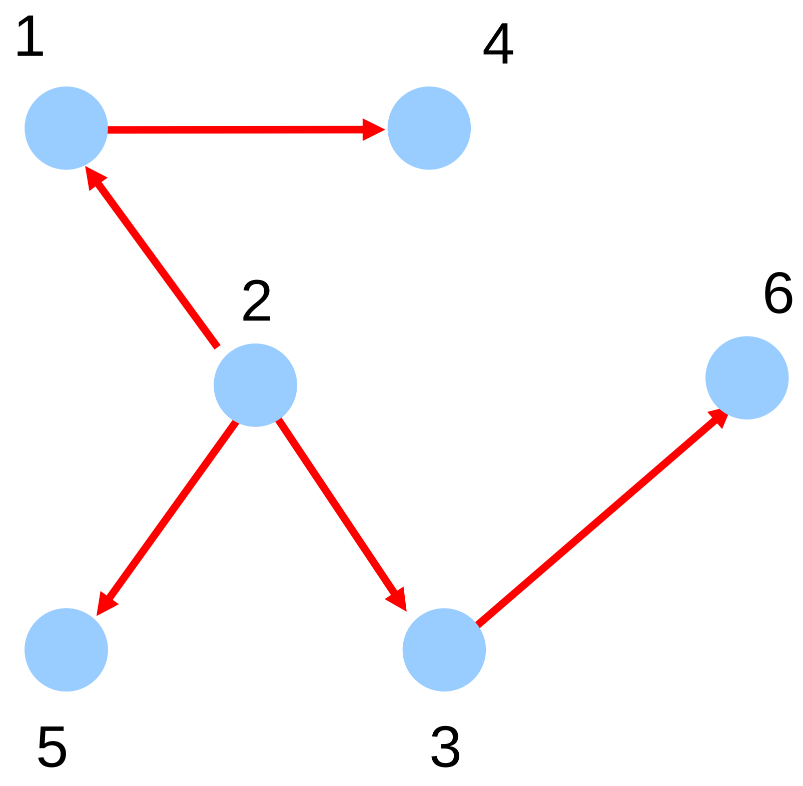
BFS – Additional Details

- What is the runtime of BFS?
  - How many times does each vertex enters the queue?
  - Each edge is considered only once.
- Therefore, the runtime of BFS should be O(m + n).

BFS – Additional Details

- The π value of a vertex v denotes the vertex u that discovered v.
- The π values maintained during BFS can be used to define a subgraph of G as follows.
- Define the predecessor subgraph of G = (V,E) as
  - $G_\pi = (V_\pi, E_\pi)$ where
  - $V_\pi = \{v \in V : \pi(v) \mathrel{!=} \text{NULL}\} \cup \{s\}$, i.e., all vertices reached during a BFS from s, and
  - $E_\pi = \{(\pi(v), v) \in E : v \in V_\pi - \{s\}\}$, directed edges from the parent of a vertex to the vertex.

BFS Example Contd...



Properties of BFS

- Consider the time at which a vertex v has entered the queue.
- The state of v at that instant changes from NOT_VISITED to IN_PROGRESS.
- $d_s(v)$ changes to a finite value, and
- $d_s(v)$ can never change after that instant.

Classifying Edges

- Can classify edges of G according to BFS from a given s as follows.
- The edges of E_π are also called as tree edges.
- It holds that for a tree edge (u, v), d(v) = d(u) + 1.
- The edges of $E_N := E \setminus E_\pi$ are called as non-tree edges.
- These edges can be further classified as follows.