CS 401: Computer Algorithm I

Graphs

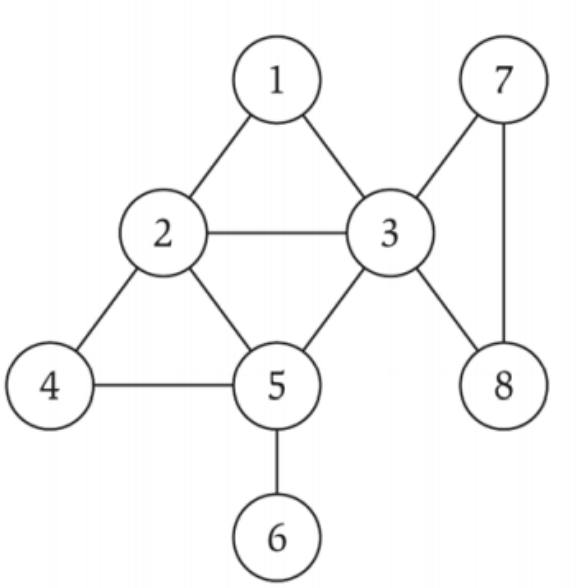
Xiaorui Sun

Graph algorithms

Undirected Graphs G=(V,E)

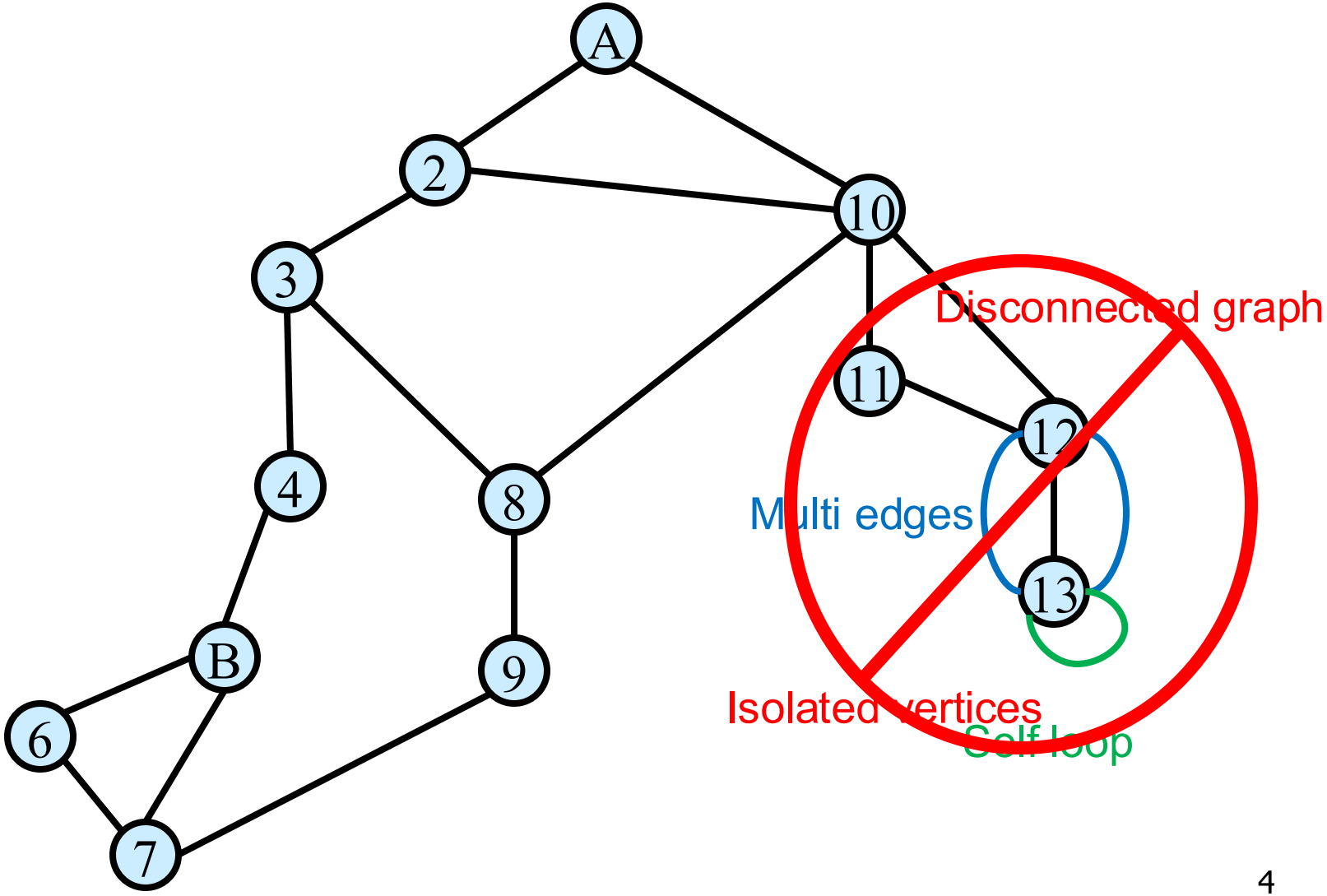
Notation. G = (V, E)

- V = nodes (or vertices)
- E = edges between pairs of nodes
- Captures pairwise relationship between objects
- Graph size parameters: n = |V|, m = |E|



V = {1, 2, 3, 4, 5 ,6, 7, 8}
E = {(1,2), (1,3), (2,3), (2,4), (2,5), (3,5), (3,7),
(3,8), (4,5), (5,6), (7,8)}
m=11, n=8

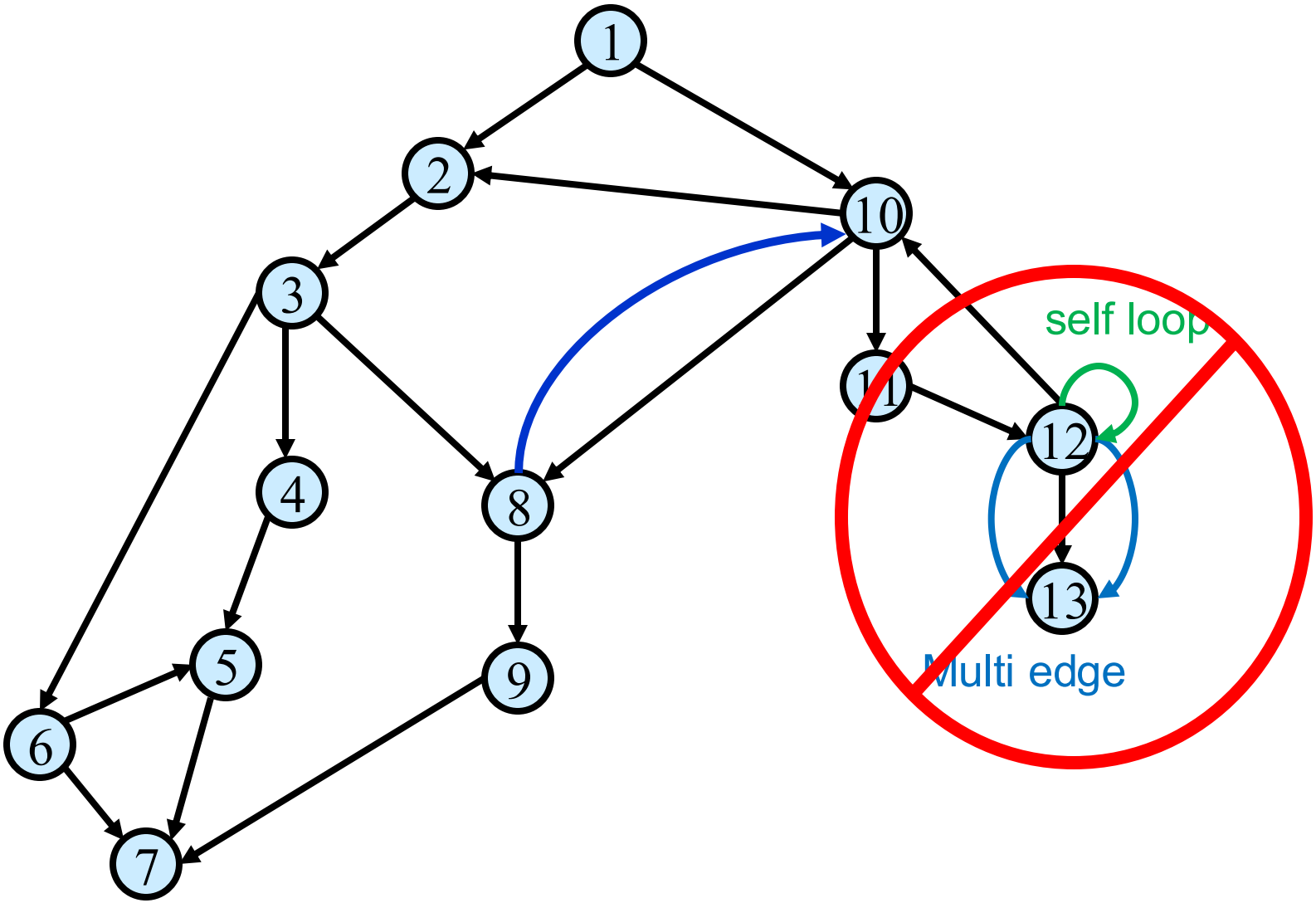
Undirected Graphs G=(V,E)



Graph applications

graph	node	edge
communication	telephone, computer	fiber optic cable
circuit	gate, register, processor	wire
mechanical	joint	rod, beam, spring
financial	stock, currency	transactions
transportation	street intersection, airport	highway, airway route
internet	class C network	connection
game	board position	legal move
social relationship	person, actor	friendship, movie cast
neural network	neuron	synapse
protein network	protein	protein-protein interaction
molecule	atom	bond

Directed Graphs



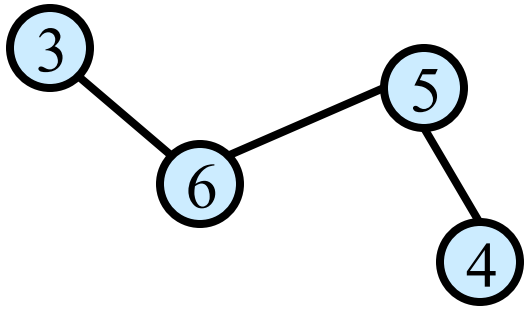
Terminology

- Path: A sequence of vertices

s.t. each vertex is connected

to the next vertex with an edge

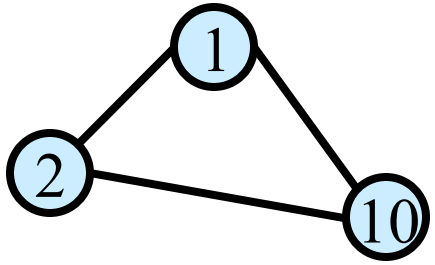
- A path is simple if all nodes are distinct



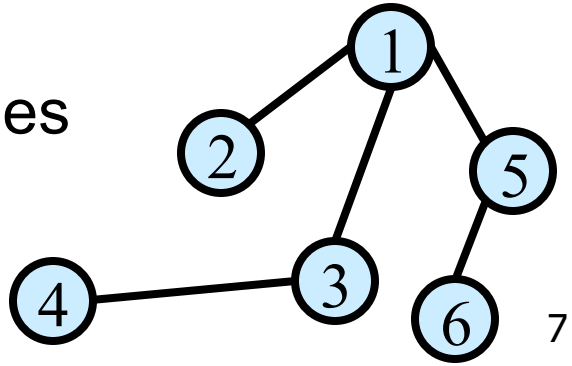
- Cycle: Path of length > 2 that has

the same start and end

- A cycle is simple if all nodes are distinct



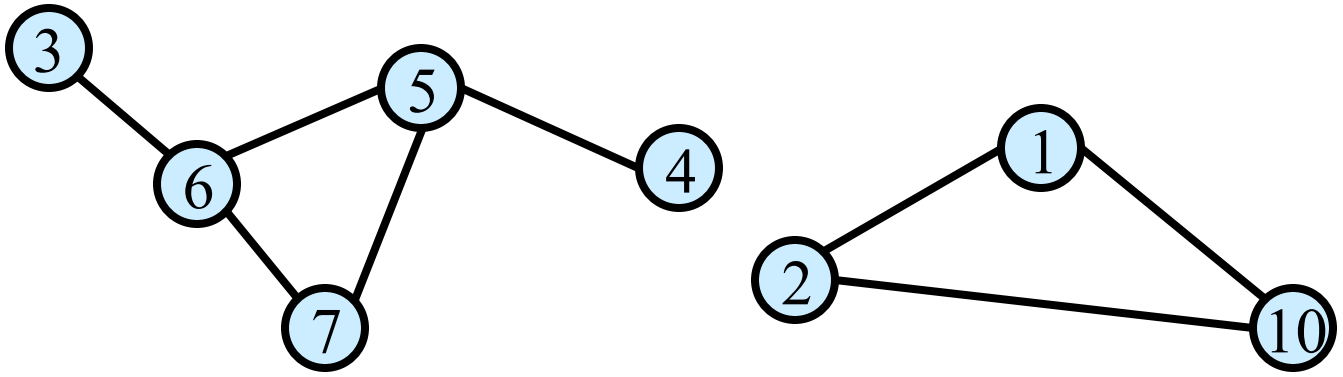
- Tree: A connected graph with no cycles



Terminology (cont'd)

- Degree of a vertex: # edges that touch that vertex

deg(6)=3



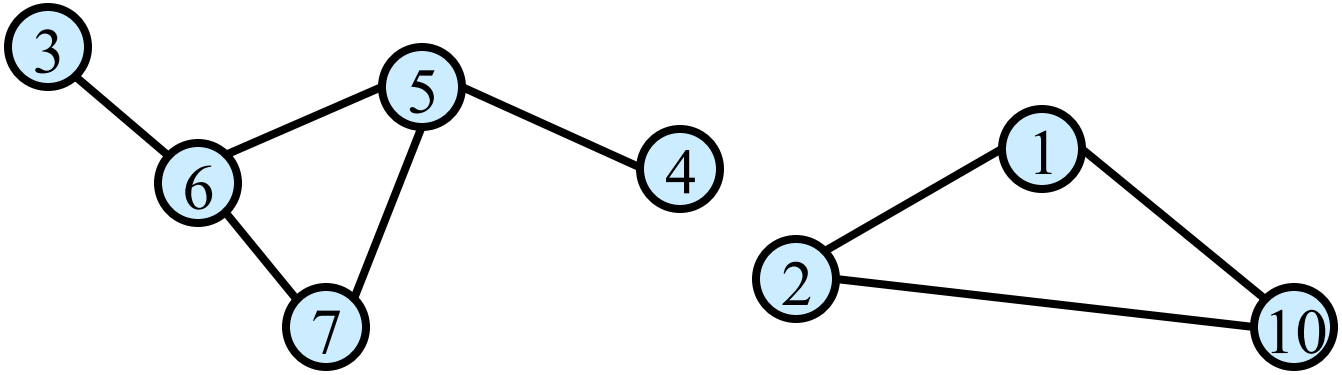
- Connected: Graph is connected if there is a path between every two vertices

- Connected component: Maximal set of connected vertices

Degree Sum

Claim: In any undirected graph, the number of edges is equal to $(1/2)\sum_{\text{vertex } v} \deg(v)$

Pf: $\sum_{\text{vertex } v} \deg(v)$ counts every edge of the graph exactly twice; once from each end of the edge.



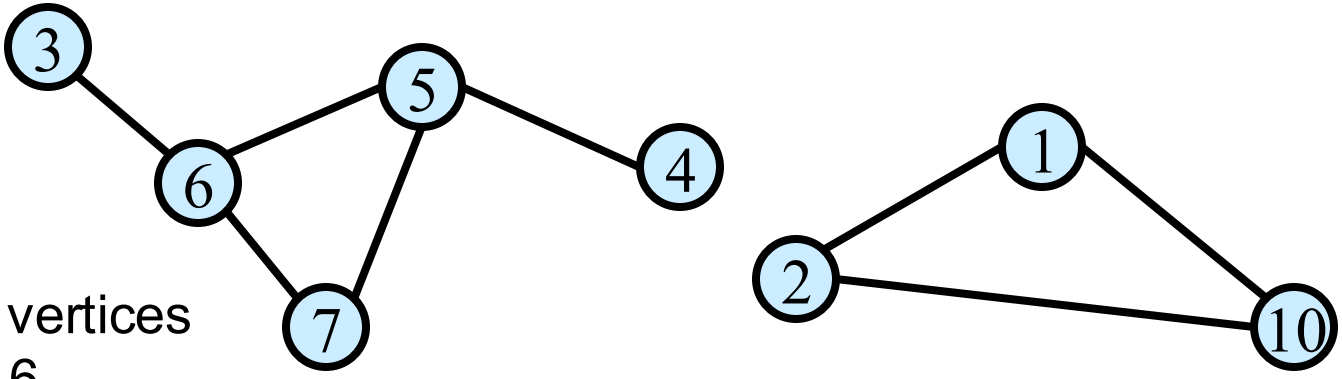
|E|=8

$$\sum_{\text{vertex } v} \deg(v) = 2 + 2 + 1 + 1 + 3 + 2 + 3 + 2 = 16$$

Odd Degree Vertices

Claim: In any undirected graph, the number of odd degree vertices is even

Pf: In previous claim we showed sum of all vertex degrees is even. So there must be even number of odd degree vertices, because sum of odd number of odd numbers is odd.



4 odd degree vertices
3, 4, 5, 6

#edges

Let $G = (V, E)$ be a graph with $n = |V|$ vertices and $m = |E|$ edges.

Claim: $0 \leq m \leq \binom{n}{2} = \frac{n(n-1)}{2} = O(n^2)$

Pf: Since every edge connects two distinct vertices (i.e., G has no loops)

and no two edges connect the same pair of vertices (i.e., G has no multi-edges)

It has at most $\binom{n}{2}$ edges.

Degree 1 vertices

Claim: If G has no cycle, then it has a vertex of degree ≤ 1

(Every tree has a leaf (degree 1 vertex in tree))

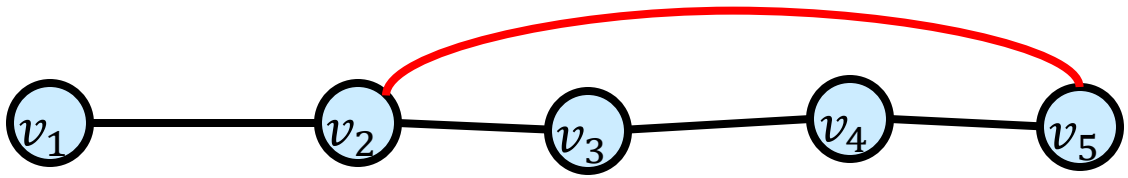
Proof: (By contradiction)

Suppose every vertex has degree ≥ 2.

Start from a vertex v_1 and follow a path, $v_1, \ldots, v_i$ when we are at v_i we choose the next vertex to be different from v_{i-1}. We can do so because $\deg(v_i) \geq 2$.

The first time that we see a repeated vertex ($v_j = v_i$) we get a cycle.

We always get a repeated vertex because G has finitely many vertices



Trees and Induction

Claim: Show that every tree with n vertices has $n - 1$ edges.

Proof: (Induction on n.)

Base Case: $n = 1$, the tree has no edge

Inductive Step: Let T be a tree with n vertices.

So, T has a vertex v of degree 1.

Remove v and the neighboring edge, and let T' be the new graph.

We claim T' is a tree: It has no cycle, and it must be connected.

So, T' has $n - 2$ edges and T has $n - 1$ edges.

Graph Traversal

Walk (via edges) from a fixed starting vertex s to all vertices reachable from s.

- Breadth First Search (BFS): Order nodes in successive layers based on distance from s
- Depth First Search (DFS): More natural approach for exploring a maze;

Applications of BFS:

- Finding shortest path for unit-length graphs
- Finding connected components of a graph
- Testing bipartiteness

Breadth First Search (BFS)

Completely explore the vertices in order of their distance from s.

Three states of vertices:

- Undiscovered
- Discovered
- Fully-explored

Naturally implemented using a queue

The queue will always have the list of Discovered vertices

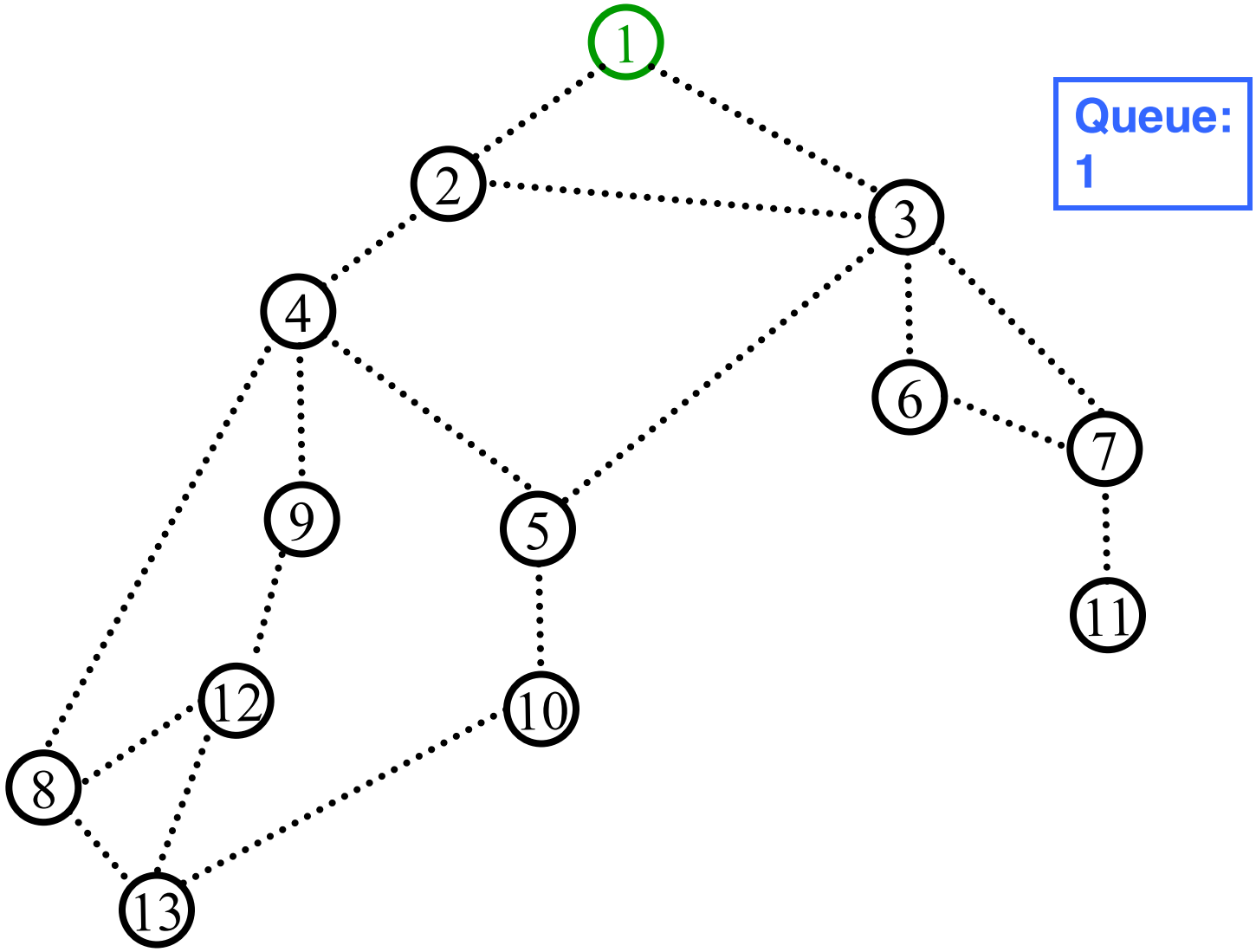
BFS algorithm

Initialization: mark all vertices "undiscovered"

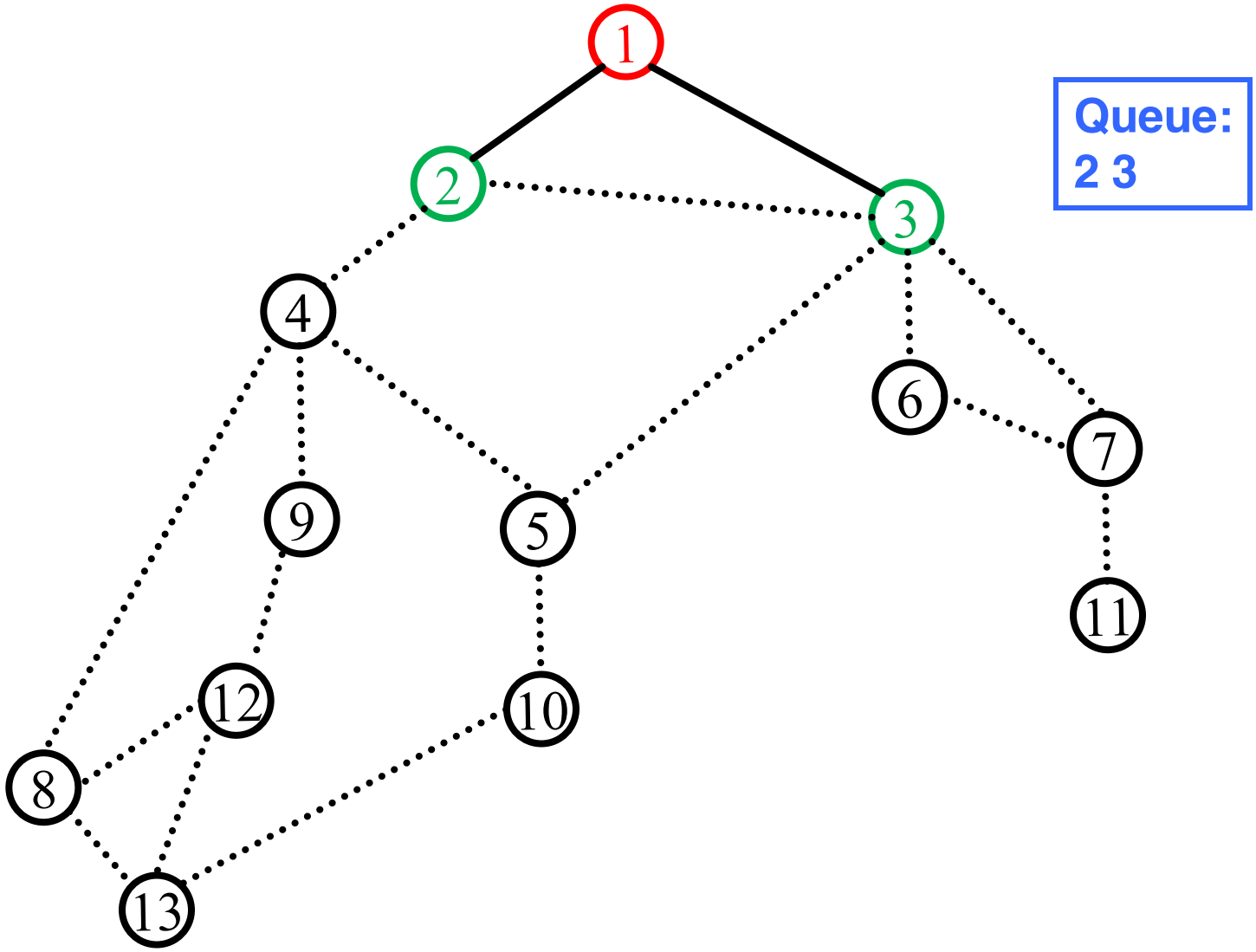
BFS(s)

    mark s discovered
    queue = { s }
    while queue not empty
        u = remove_first(queue)
        for each edge $\{u, x\}$
            if (x is undiscovered)
                mark x discovered
                append x on queue
        mark u fully-explored

BFS(1)

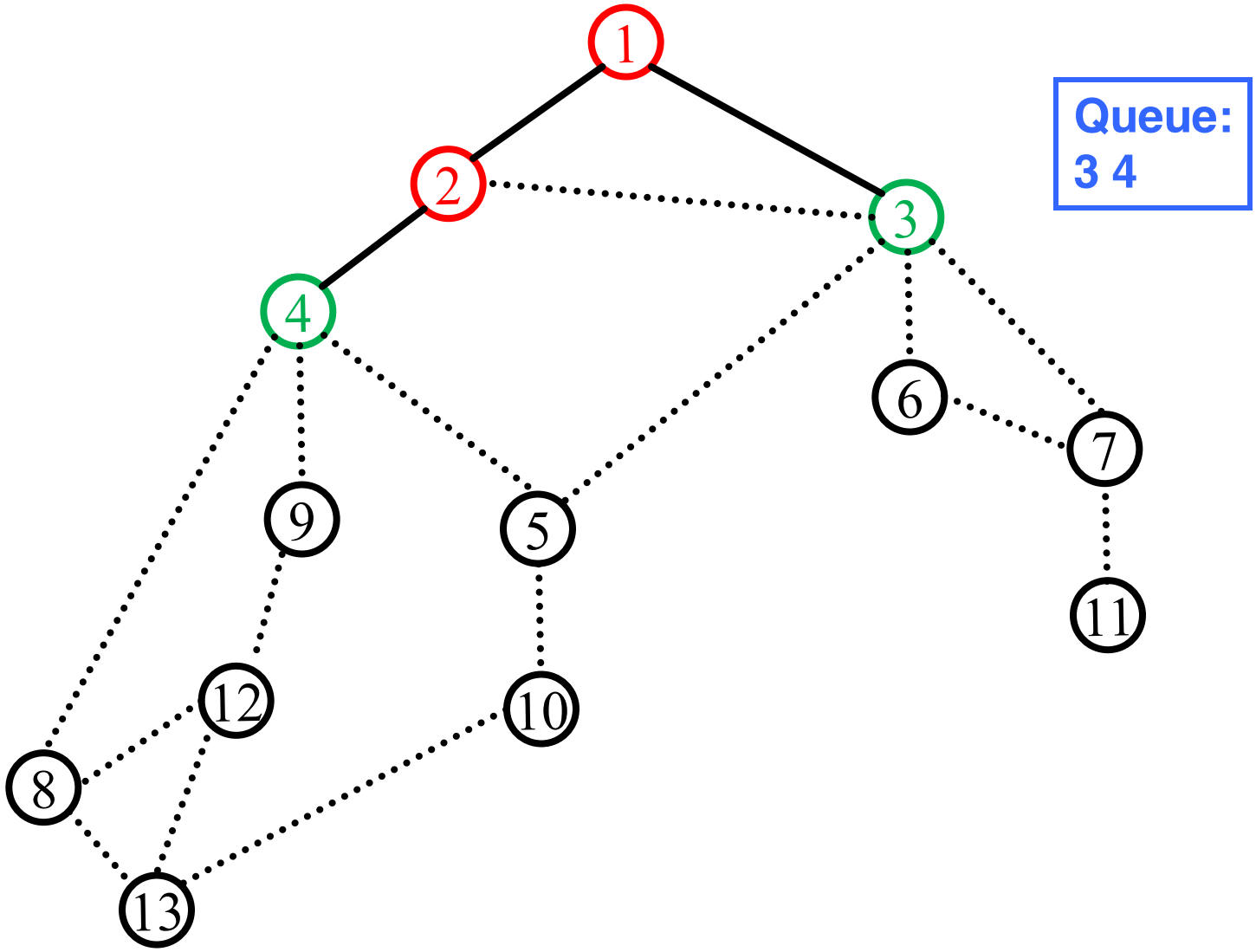


BFS(1)

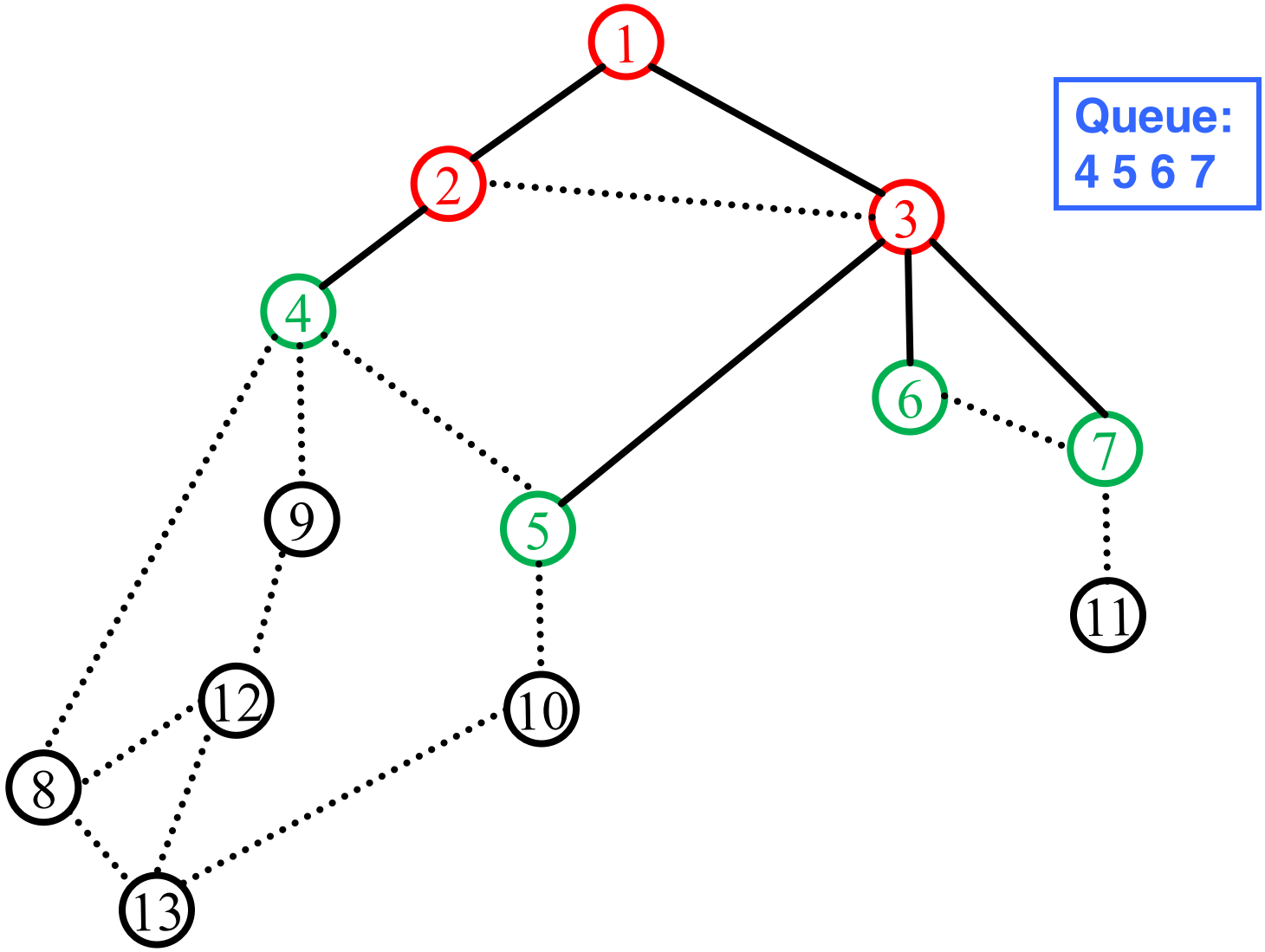


Queue:
2 3

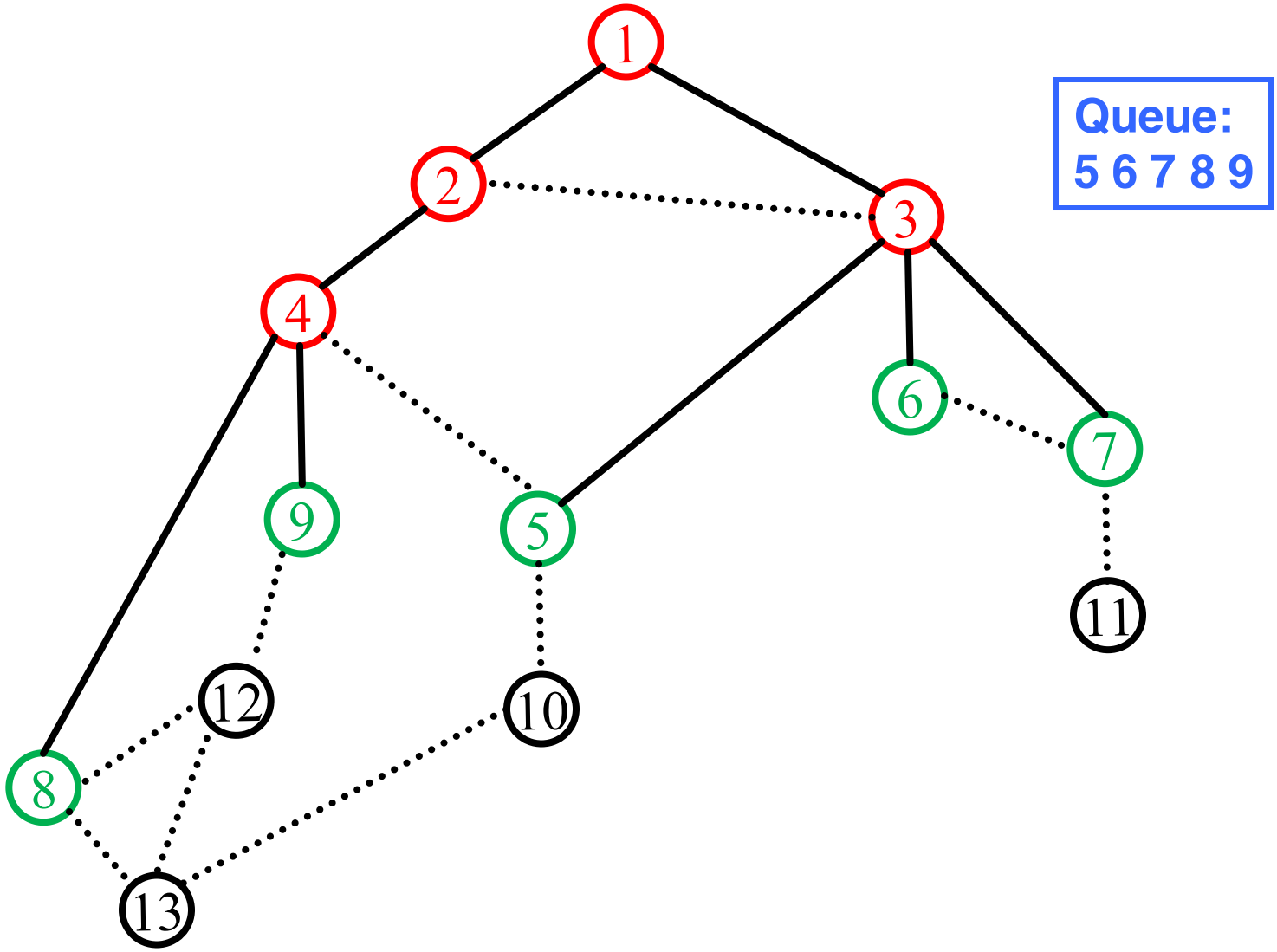
BFS(1)



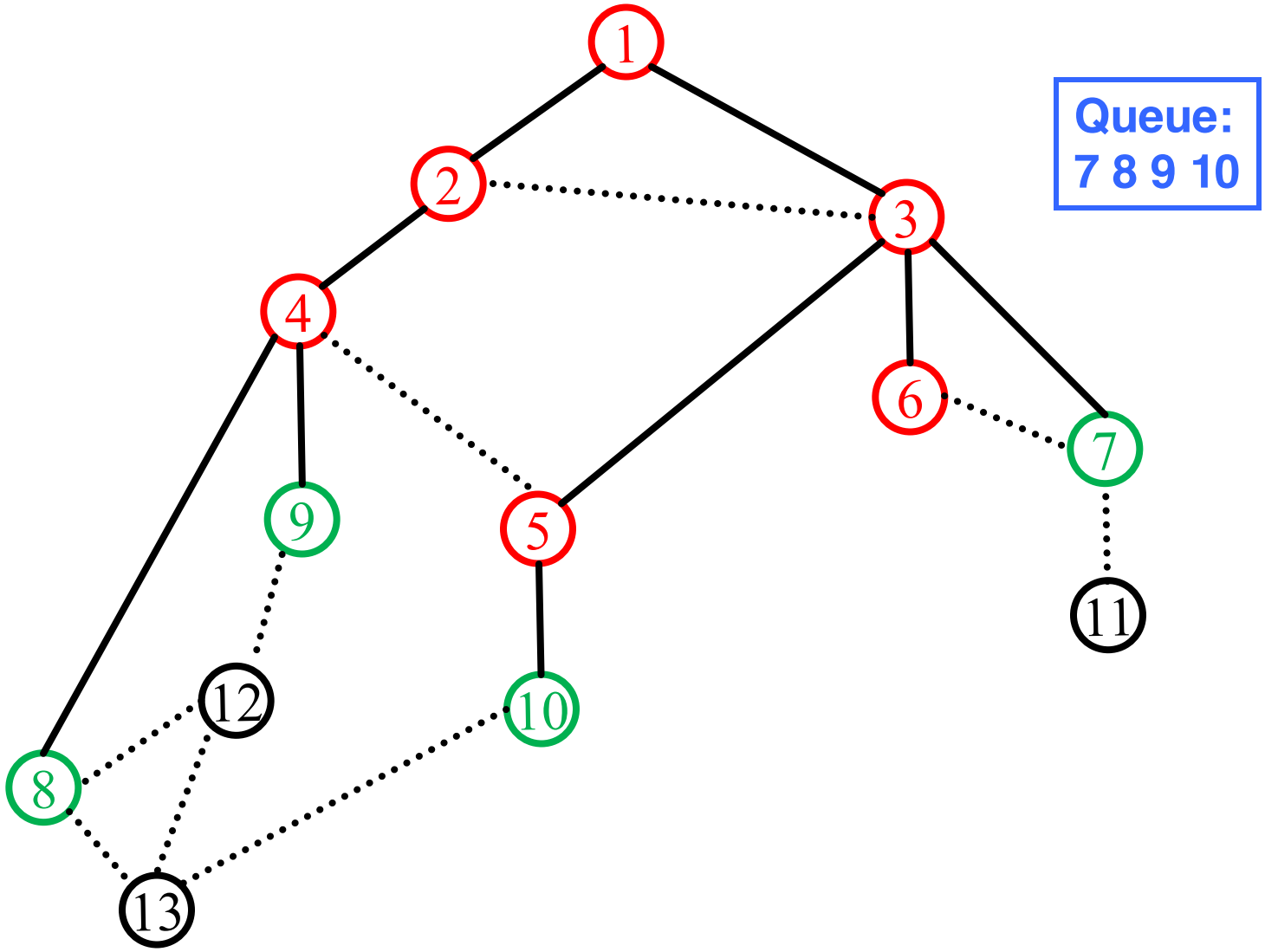
BFS(1)



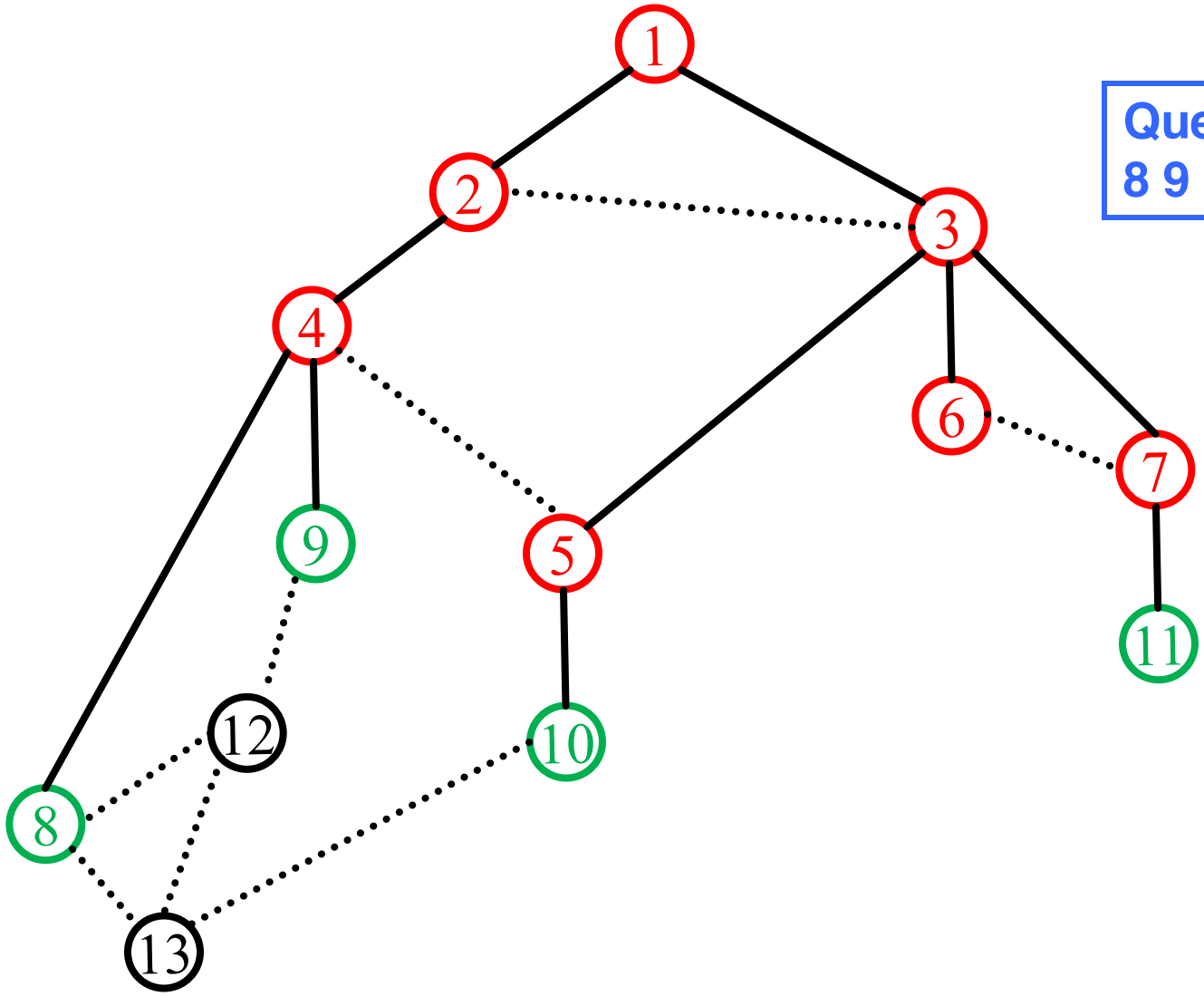
BFS(1)



BFS(1)

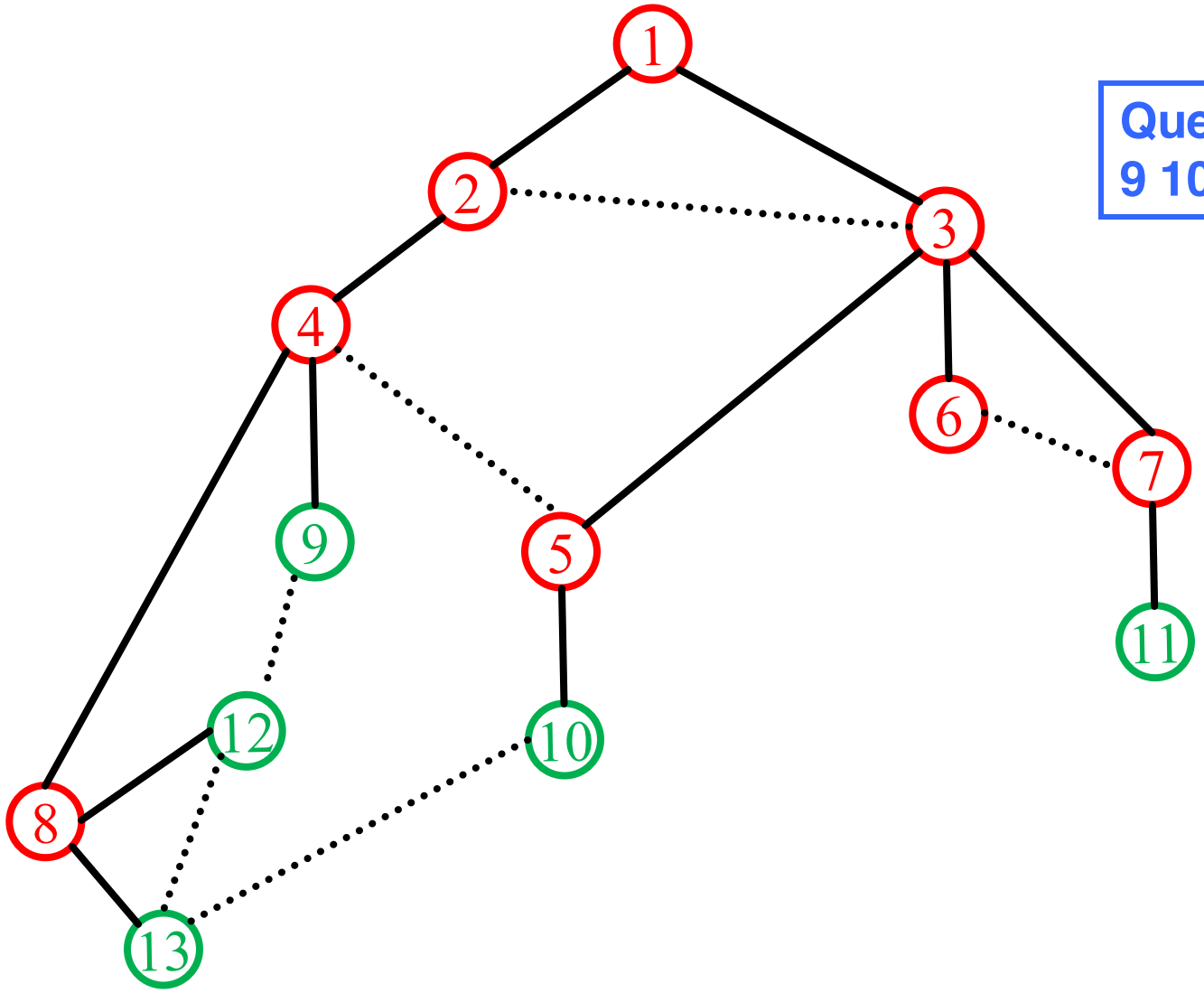


BFS(1)



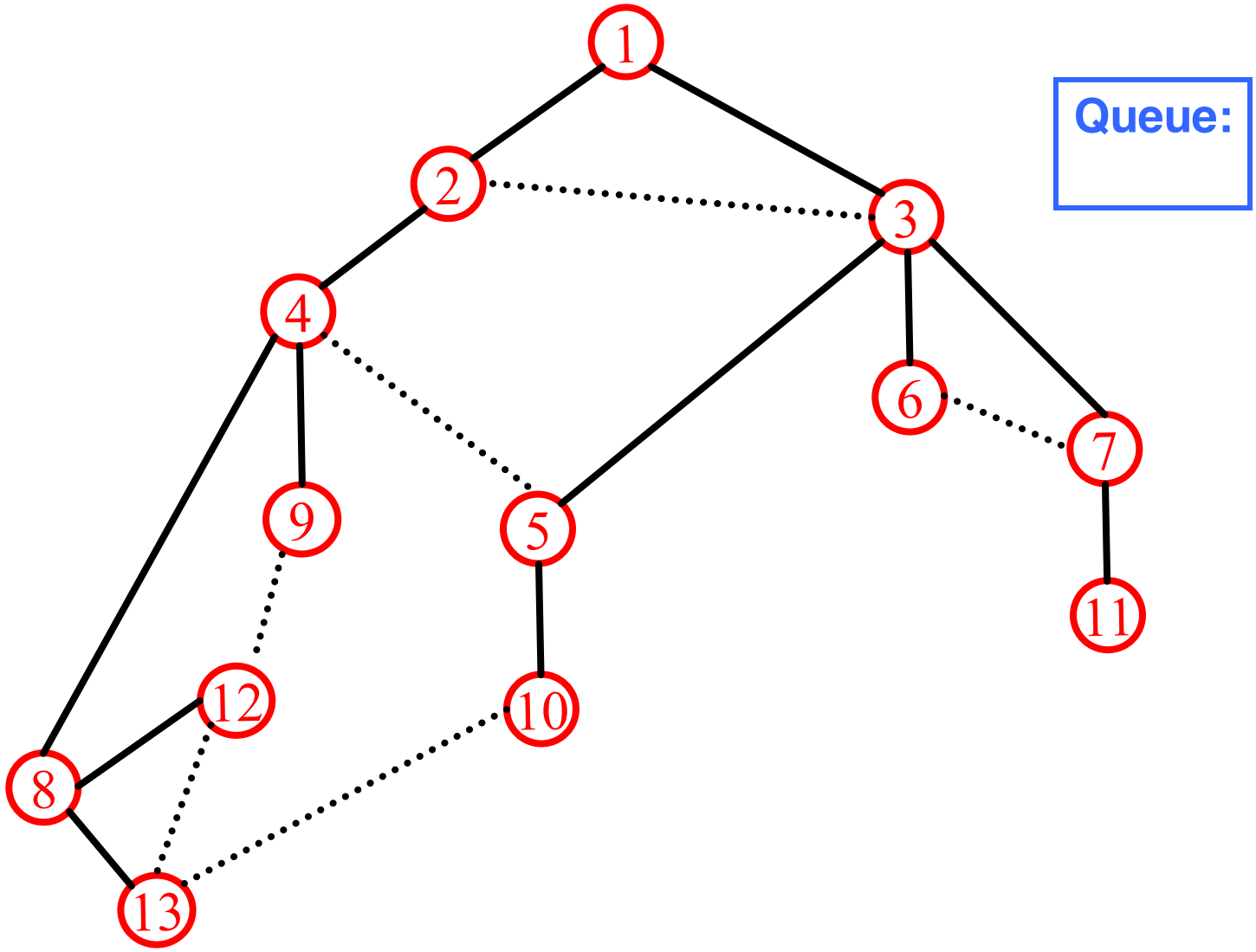
Queue:
8 9 10 11

BFS(1)



Queue:
9 10 11 12 13

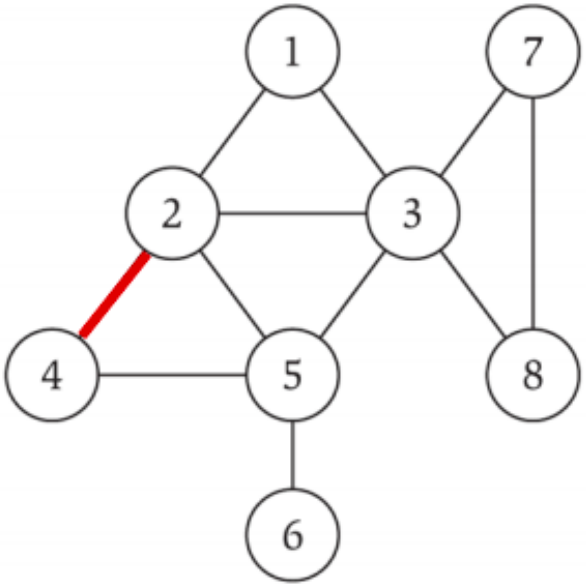
BFS(1)



Graph representation

Adjacency matrix. n-by-n matrix with $A_{uv} = 1$ if (u, v) is an edge.

- Space proportional to n^2.
- Checking if (u, v) is an edge takes $\Theta(1)$ time.
- Identifying all edges takes $\Theta(n^2)$ time.



	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

BFS Analysis

Initialization: mark all vertices "undiscovered"

Graph representation: adjacency matrix

BFS(s)

mark s discovered

queue = { s }

O(n) times:
At most once per vertex

while queue not empty

u = remove_first(queue)

O(n) times:
Check every vertex x

for each edge $\{u, x\}$

if (x is undiscovered)

mark x discovered

append x on queue

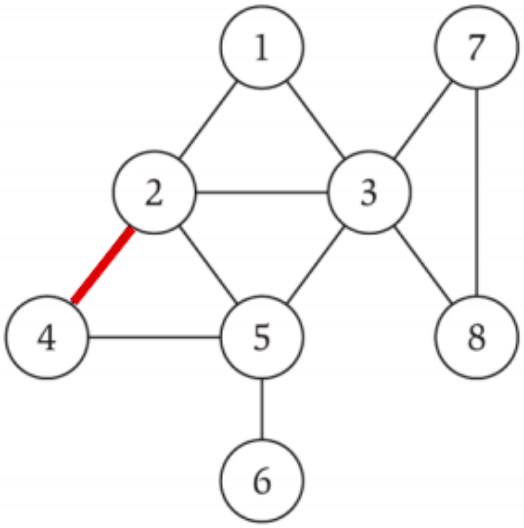
mark u fully-explored

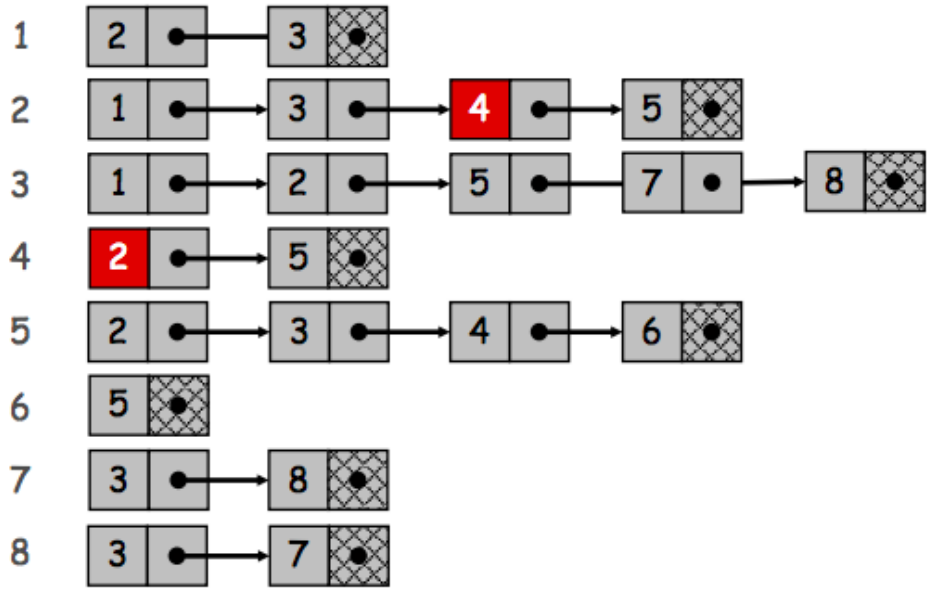
Overall: $O(n^2)$ time

Graph representation

Adjacency list. Node indexed array of lists.

- Space proportional to m+n.
- Checking if (u, v) is an edge takes O(deg(u)) time.
- Identifying all edges takes Θ(m+n) time.





BFS Analysis

Initialization: mark all vertices "undiscovered"

Graph representation: adjacency list

BFS(s)

mark s discovered

queue = { s }

O(n) times:
At most once per vertex

while queue not empty

u = remove_first(queue)

O(deg(u)) times:
At most twice per edge

for each edge $\{u, x\}$

if (x is undiscovered)

mark x discovered

append x on queue

mark u fully-explored

Overall: O(n+m) time