Introduction to Algorithms

Induction - Graphs

Degree 1 vertices

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

(So, every tree has a leaf)
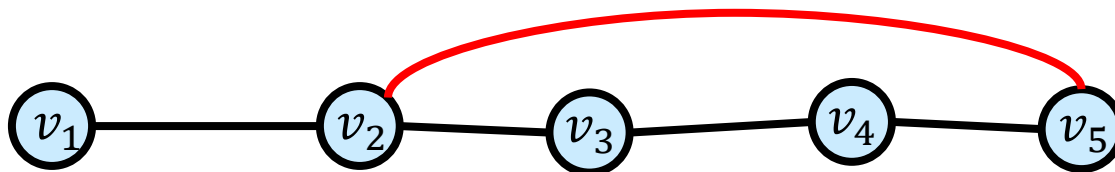
Pf: (By contradiction)

Suppose every vertex has degree ≥ 2.

Start from a vertex $!$ and follow a path, $v_1, \dots, v_i$ 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.

Pf: By induction.

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

IH: Suppose every tree with n-1 vertices has n-2 edges

IS: 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.

#edges

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

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

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.

Sparse Graphs

A graph is called sparse if $m \ll n^2$ and it is called dense otherwise.

Sparse graphs are very common in practice

- Friendships in social network
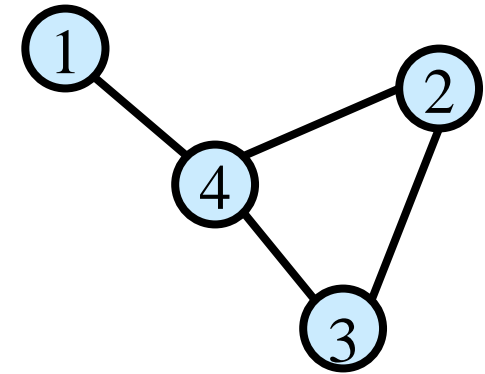- Planar graphs
- Web graph

Q: Which is a better running time $O(\text{n} + \text{m})$ vs $O(n^2)$?

A: $O(\text{n} + \text{m}) = O(n^2)$, but $O(\text{n} + \text{m})$ is usually much better.

Storing Graphs (Internally in ALG)

Vertex set $V = \{v_1, \dots, v_n\}$

Adjacency Matrix: A

- For all, i, j, $A[i, j] = 1$ if $(v_i, v_j) \in E$
- Storage: n^2 bits

Advantage:

- $O(1)$ test for presence or absence of edges

Disadvantage:

- Inefficient for sparse graphs both in storage and edge-access

	1	2	3	4
1	0	0	0	1
2	0	0	1	1
3	0	1	0	1
4	1	1	1	0

Storing Graphs (Internally in ALG)
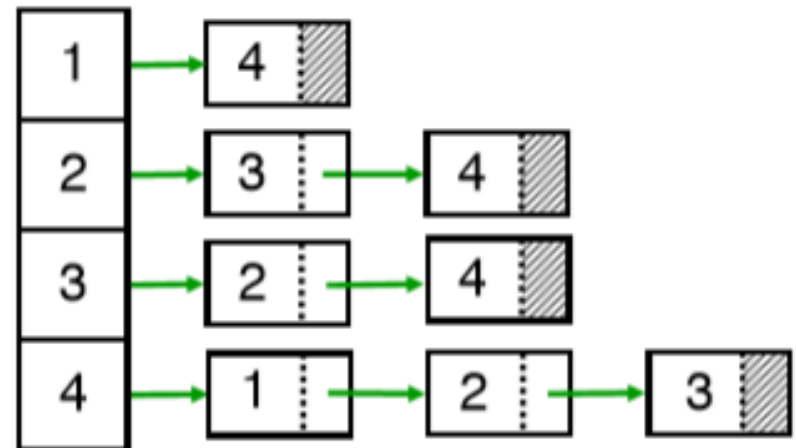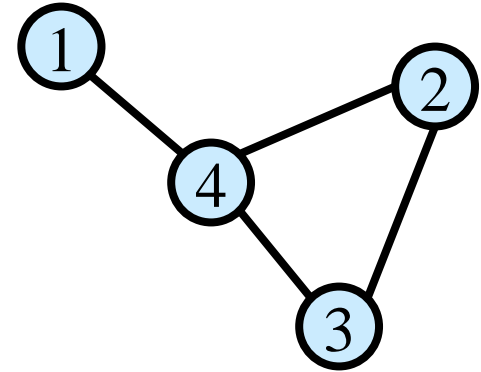
Adjacency List:

O(n+m) words

Advantage

- Compact for sparse
- Easily see all edges

Disadvantage

- No O(1) edge test
- More complex data structure

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; many efficient algs build on it.

Applications:

- Finding Connected components of a graph

- Testing Bipartiteness

- Finding Aritculation points

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 implementation

Global 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
```
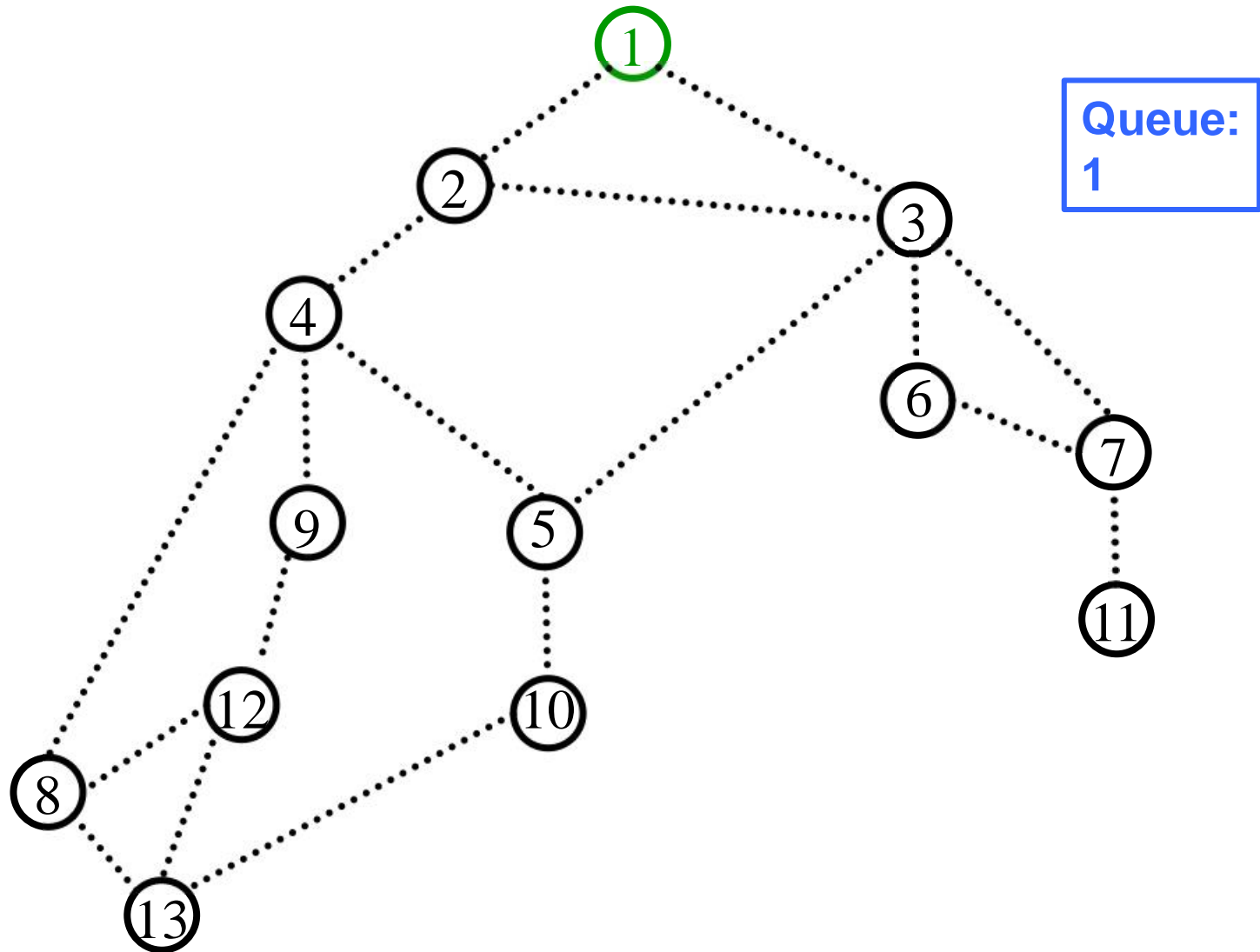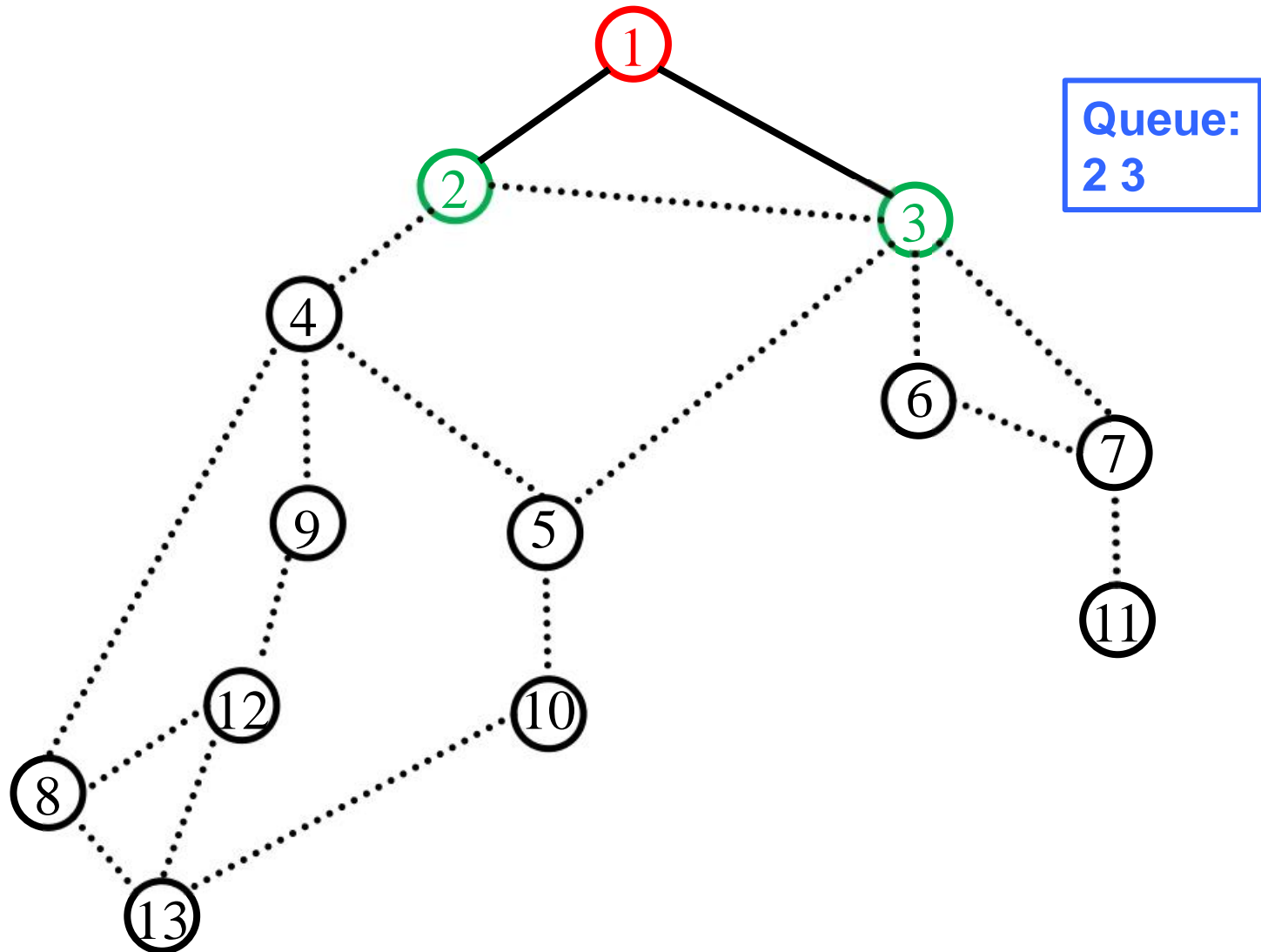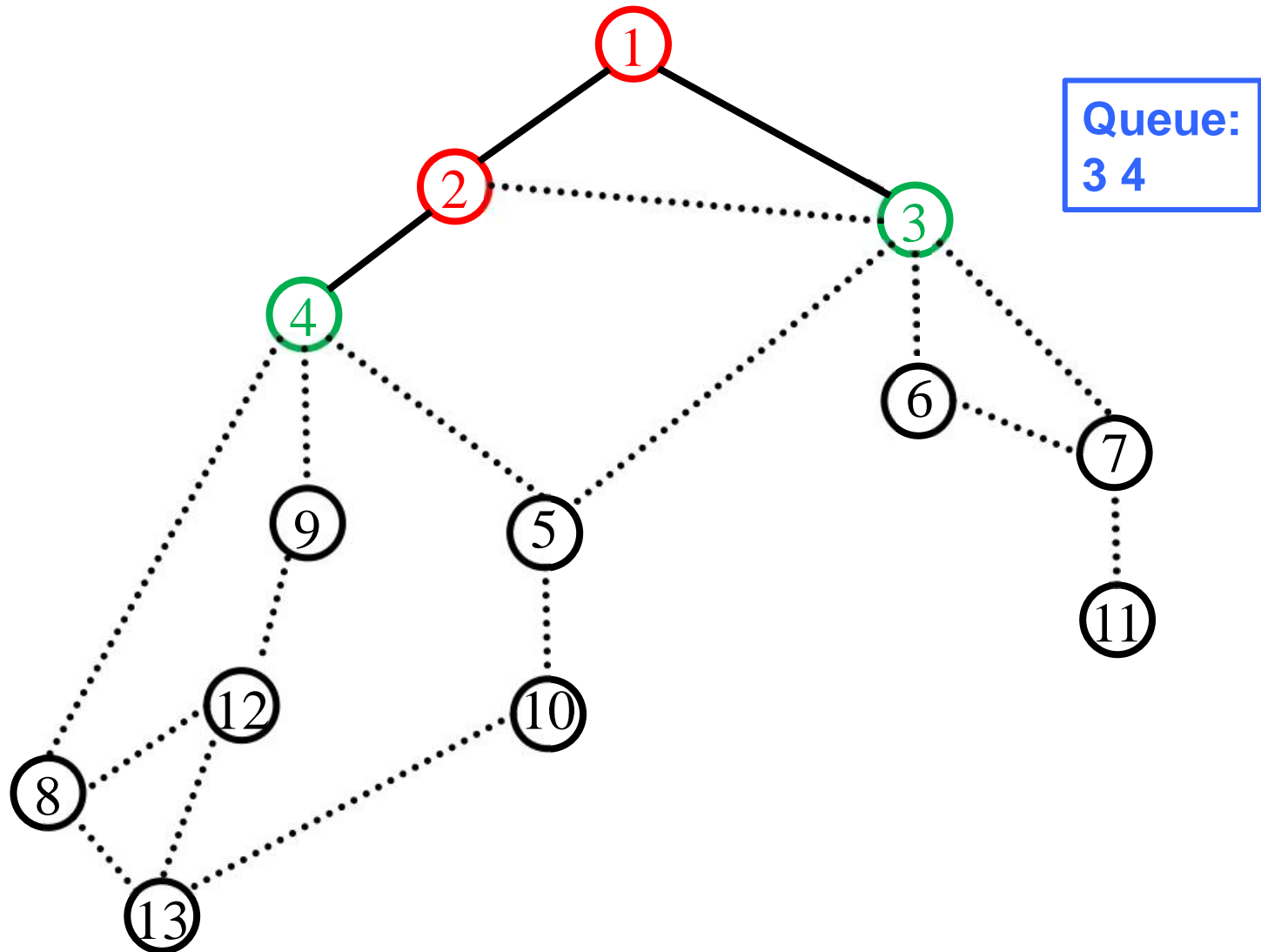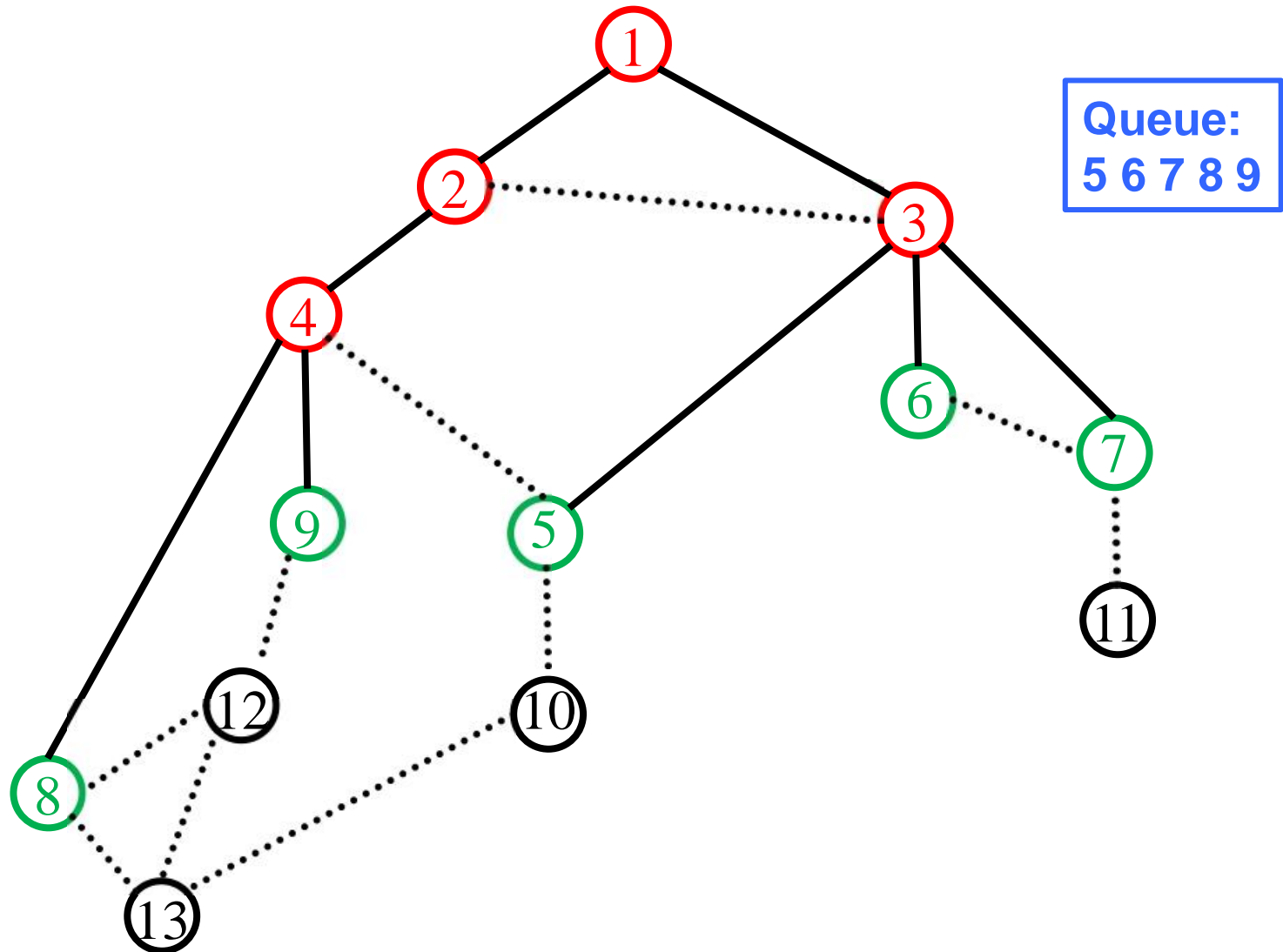
BFS(1)

Queue:
1

13

BFS(1)

Queue:
2 3

14

BFS(1)

Queue:
3 4

15

BFS(1)

Queue:
4 5 6 7

16

BFS(1)

Queue:
5 6 7 8 9

17

BFS(1)

Queue:
7 8 9 10

18

BFS(1)

Queue:
8 9 10 11

19

BFS(1)

Queue:
9 10 11 12 13

20
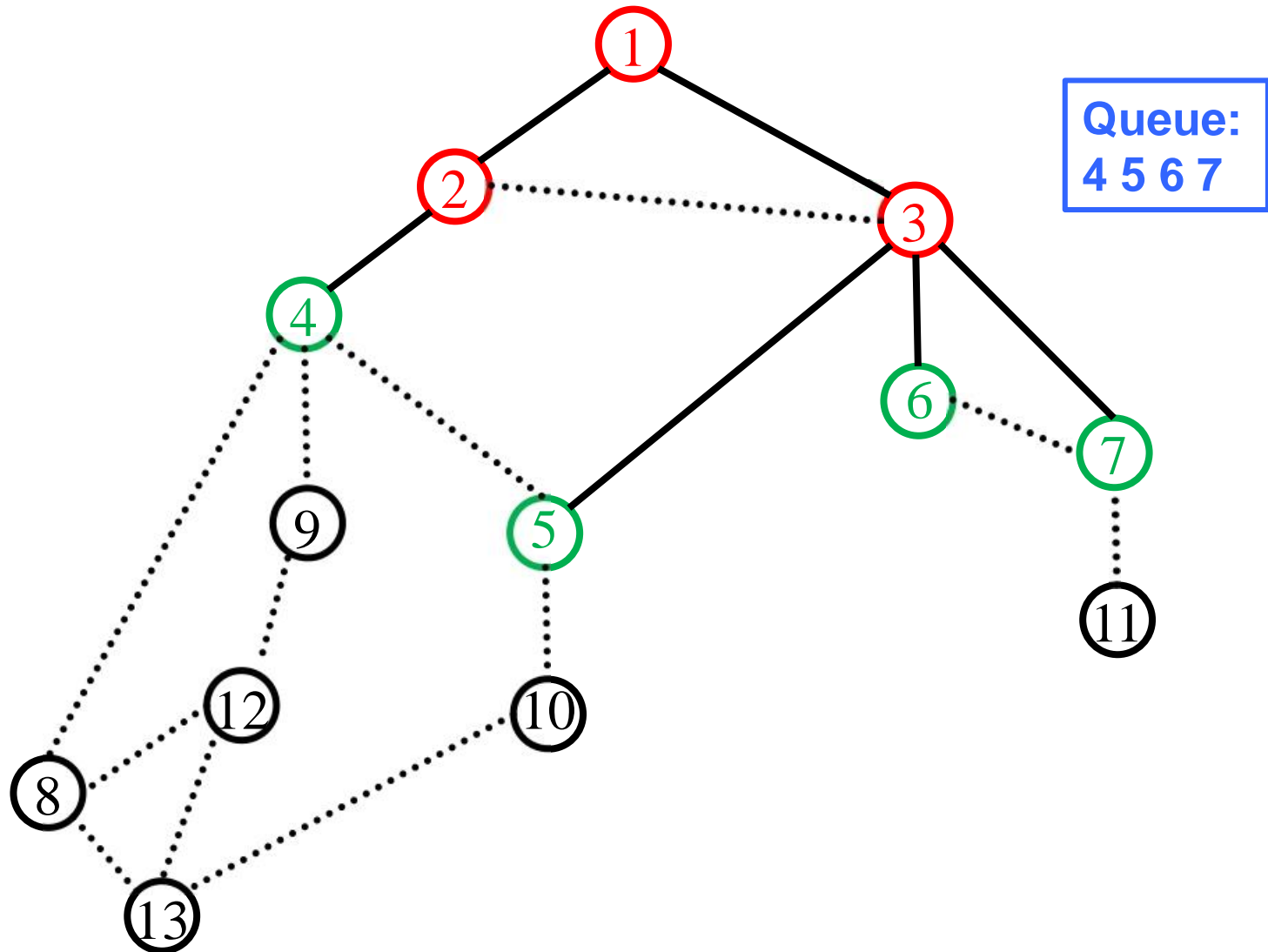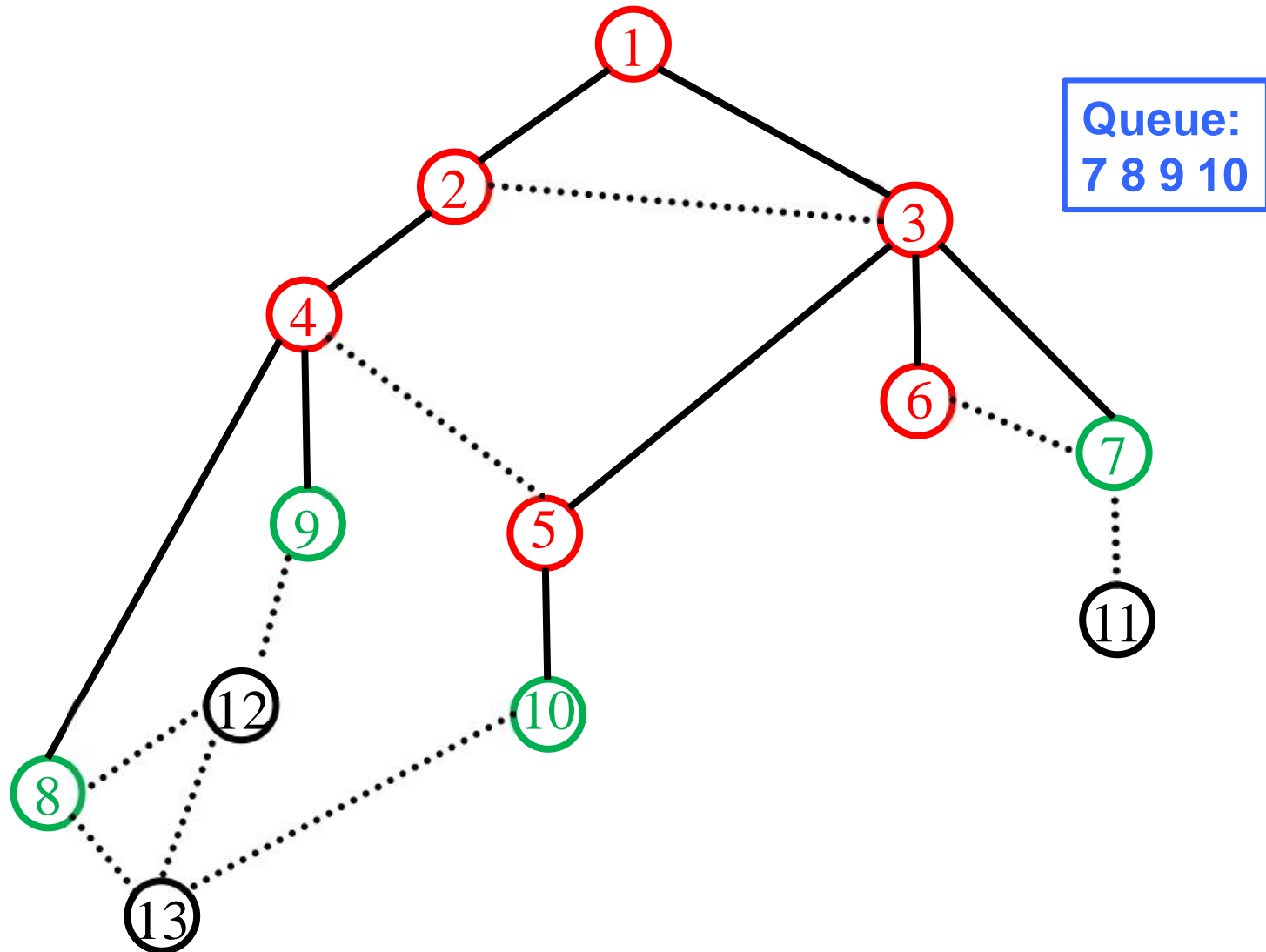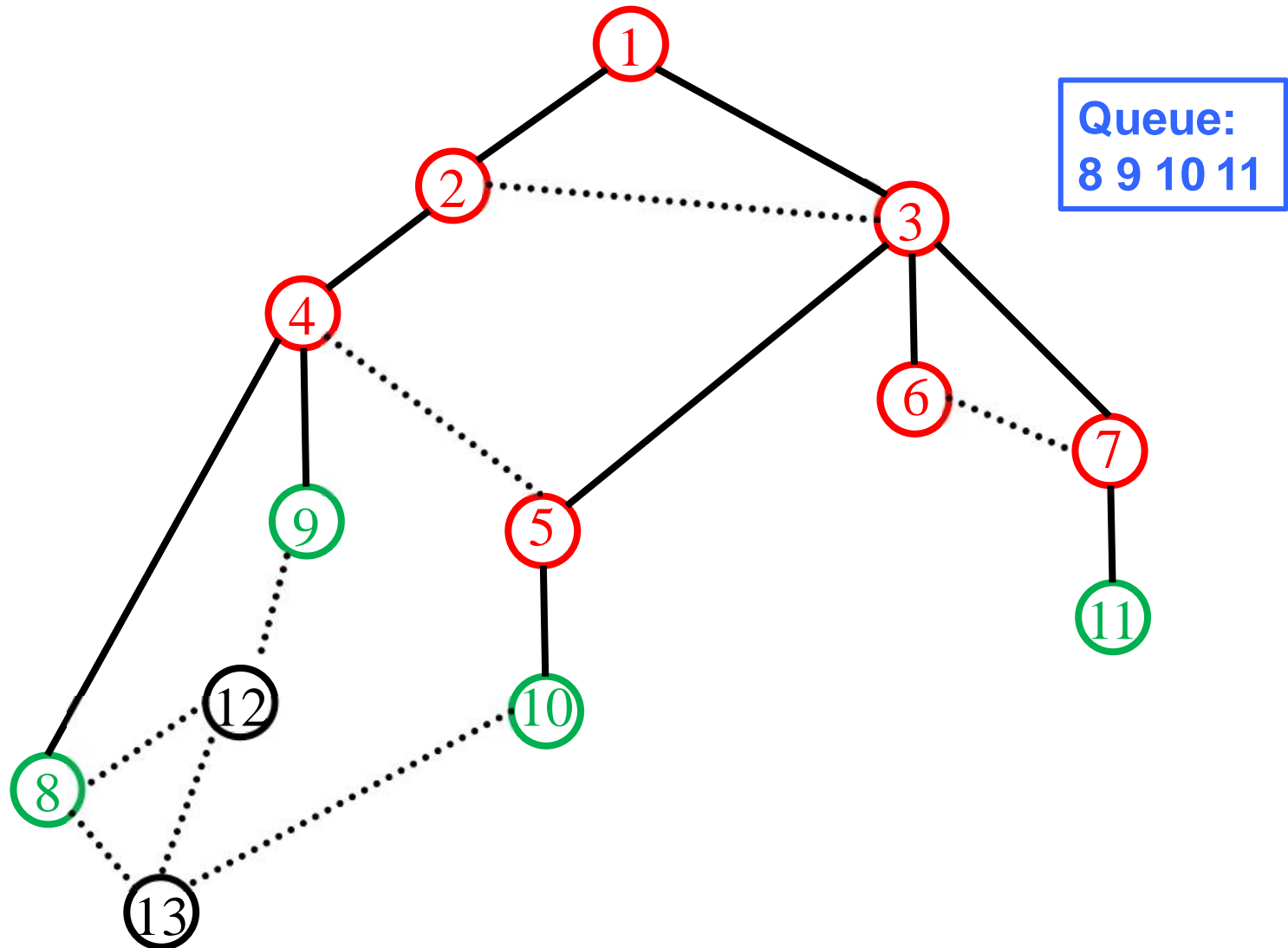
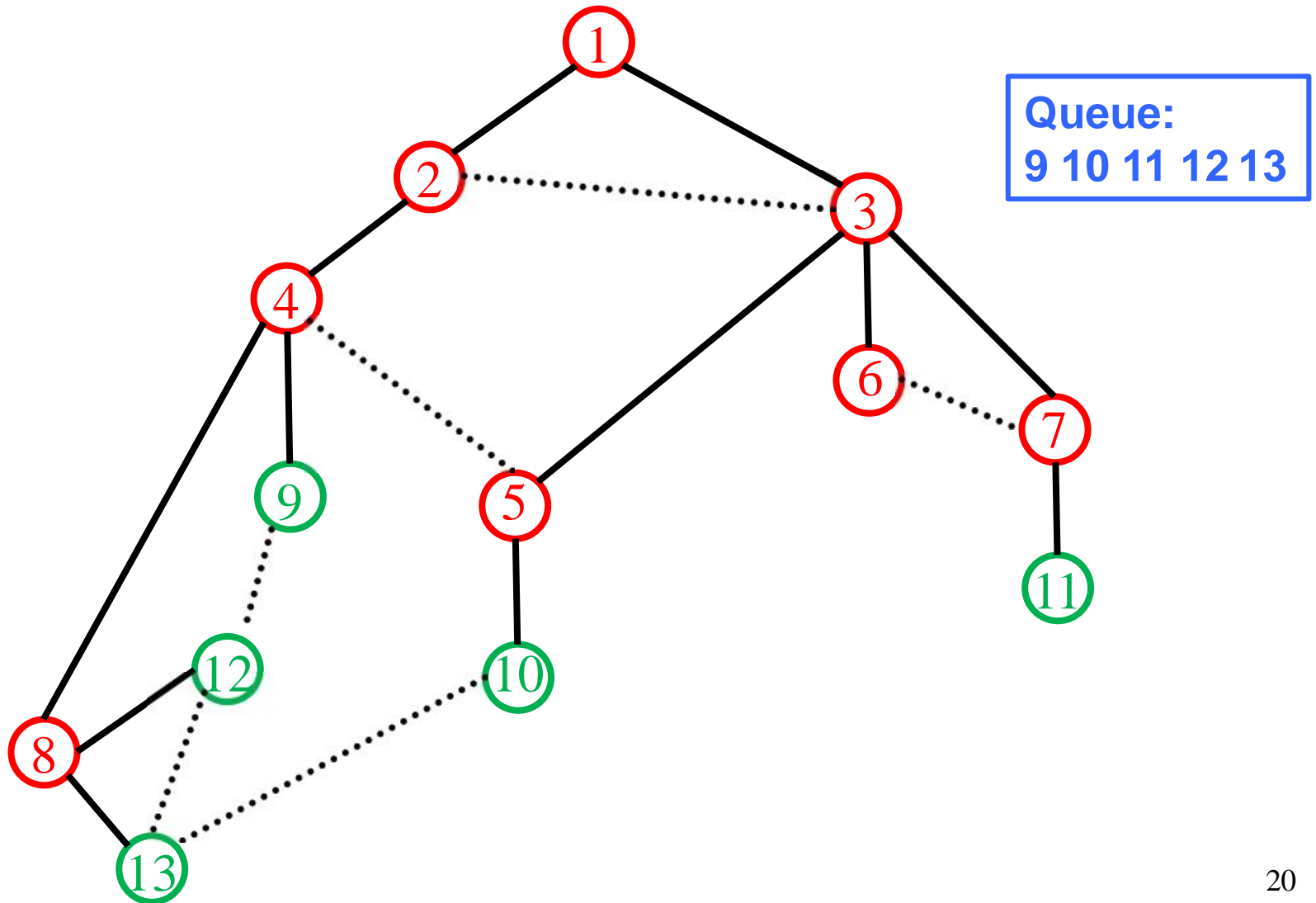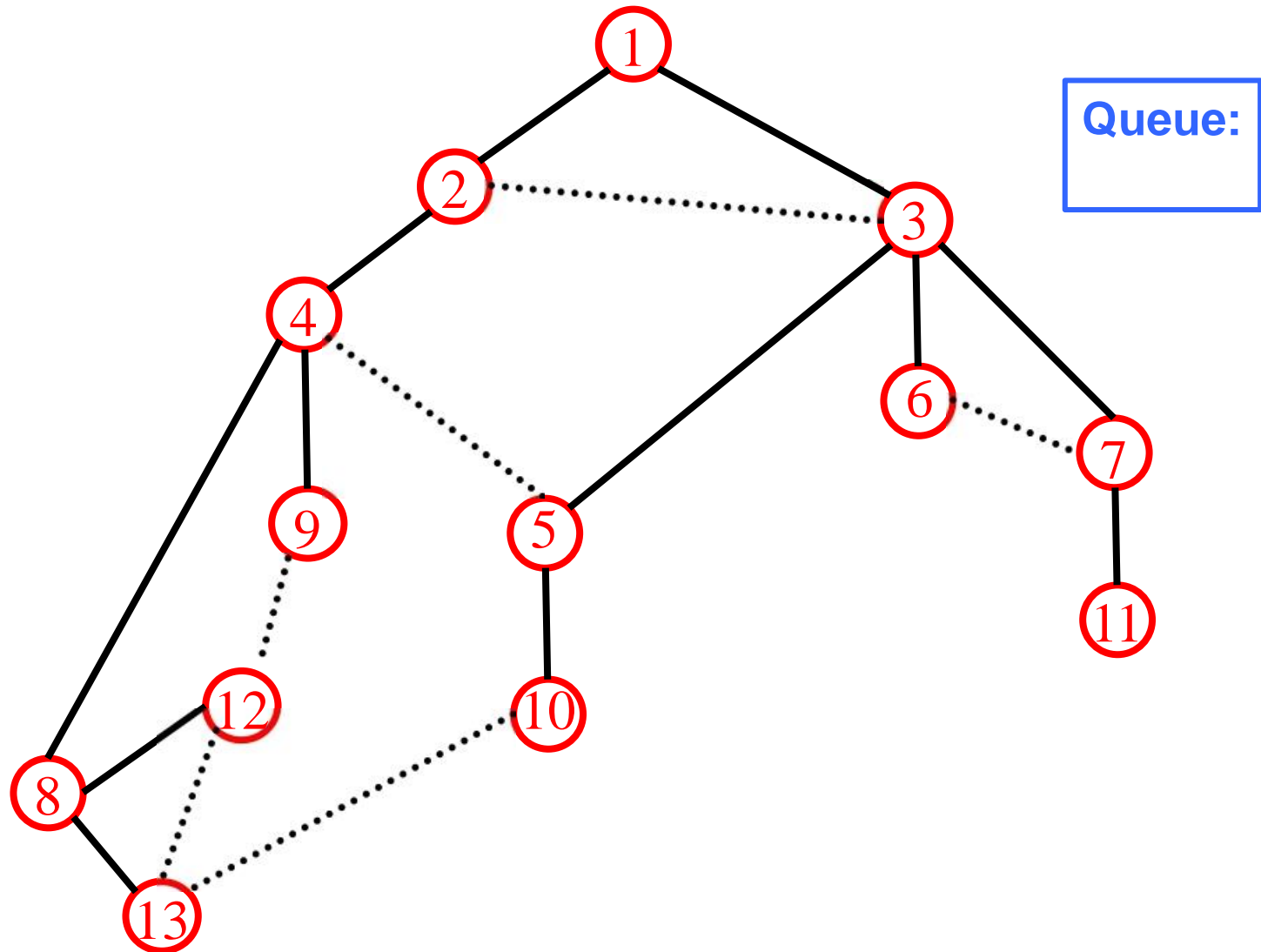# BFS(1)

Queue:

21

BFS Analysis

Global 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

O(n) times: Once from every vertex if G is connected

$deg(u) \leq O(n) \, times$

If we use adjacency list: $O(n) + O(\sum_v \deg(v)) = O(\text{n} + \text{m})$ 22

Properties of BFS

- BFS(s) visits a vertex v if and only if there is a path from s to v

- Edges into then-undiscovered vertices define a tree – the "Breadth First spanning tree" of G

- Level i in the tree are exactly all vertices v s.t., the shortest path (in G) from the root s to v is of length i

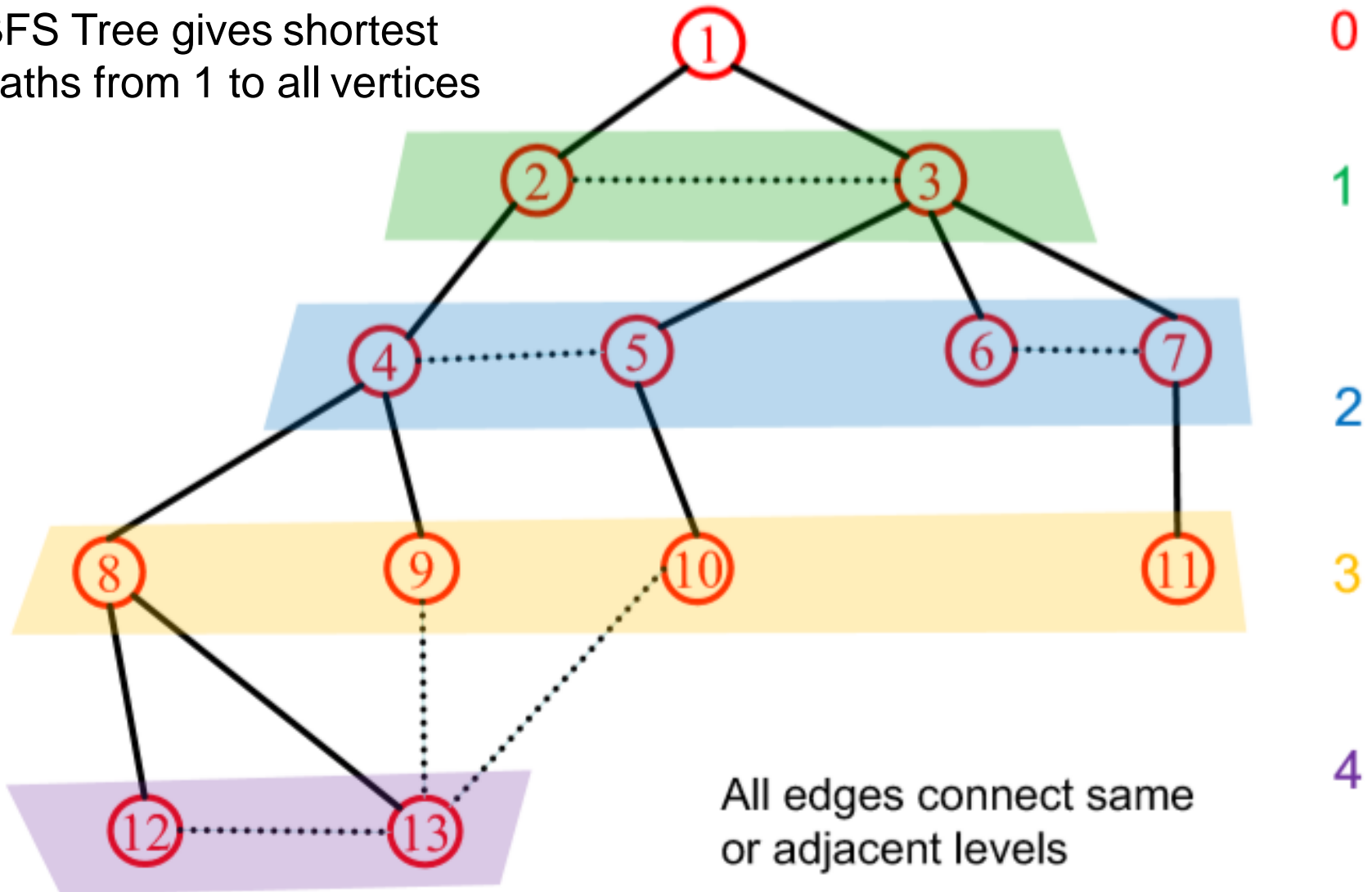- All nontree edges join vertices on the same or adjacent levels of the tree

BFS Application: Shortest Paths
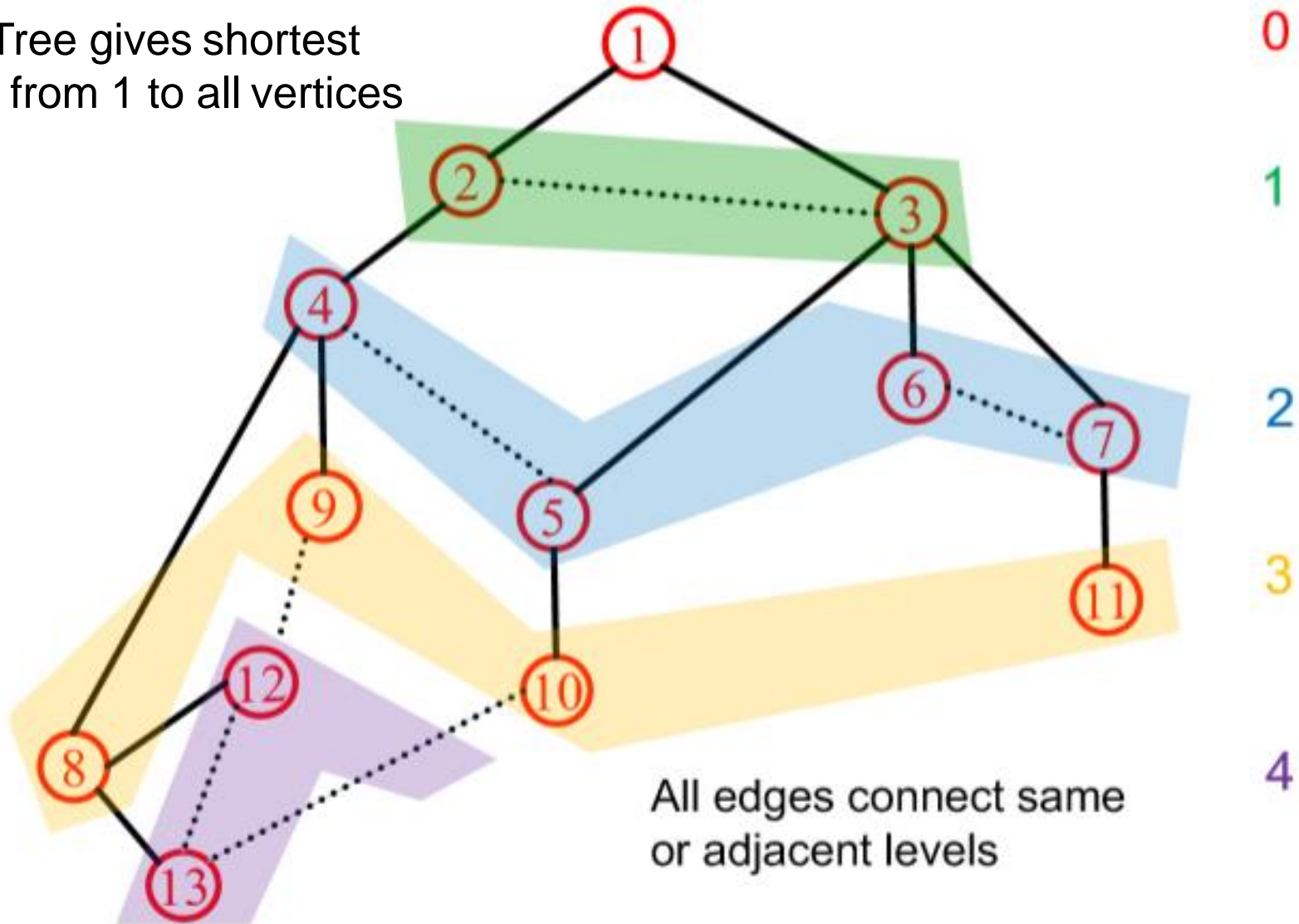
BFS Tree gives shortest
paths from 1 to all vertices

All edges connect same
or adjacent levels

24

BFS Application: Shortest Paths

BFS Tree gives shortest paths from 1 to all vertices

All edges connect same or adjacent levels

25

Properties of BFS

Claim: All nontree edges join vertices on the same or adjacent levels of the tree

Pf: Consider an edge {x,y}
Say x is first discovered and it is added to level i .
We show y will be at level i or $i + 1$

This is because when vertices incident to x are considered in the loop, if y is still undiscovered, it will be discovered and added to level $i + 1$.

Properties of BFS

Lemma: All vertices at level i of BFS(s) have shortest path distance i to s.

Claim: If $L(v) = i$ then shortest path $\leq i$

Pf: Because there is a path of length i from s to v in the BFS tree

Claim: If shortest path = i then $L(v) \leq i$

Pf: If shortest path = i, then say $s = v_0, v_1, \dots, v_i = v$ is the shortest path to v.
By previous claim,

$$L(v_1) \leq L(v_0) + 1$$
$$L(v_2) \leq L(v_1) + 1$$
$$\dots$$
$$L(v_i) \leq L(v_{i-1}) + 1$$

So, $L(v_i) \leq i$.

This proves the lemma.