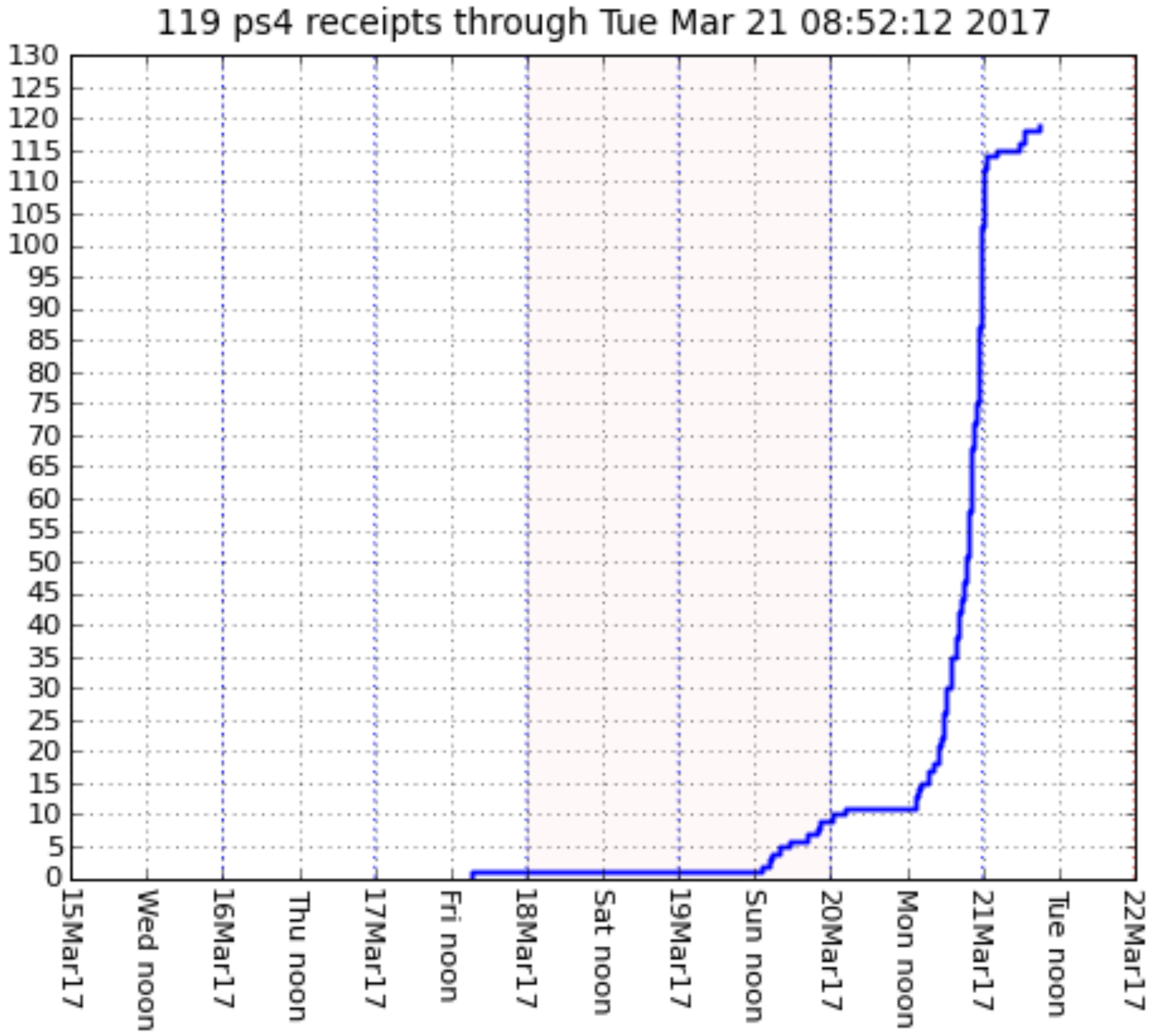
Info 2950, Lecture 15

21 Mar 2017

Prob Set 4: due Mon night 20 Mar

Midterm: Thu 23 Mar.
Bring fully charged laptop

counting / probability
bayes theorem / naive bayes classifier,
normal and poisson distributions.
simple python



(open notebook / open computer, so you'll have everything available.)

Ideally if you've done a conscientious job and understand everything on the probsets, then doesn't require much concentrated studying, but useful to have an idea of where everything is to make it easier to find in real time during exam.
On probsets so far, ps2 3,6; ps3 1,2,3; ps4 1,2 ;
are either adapted from or similar to past midterm problems.

programming minimal at most, but bring charged laptop ...

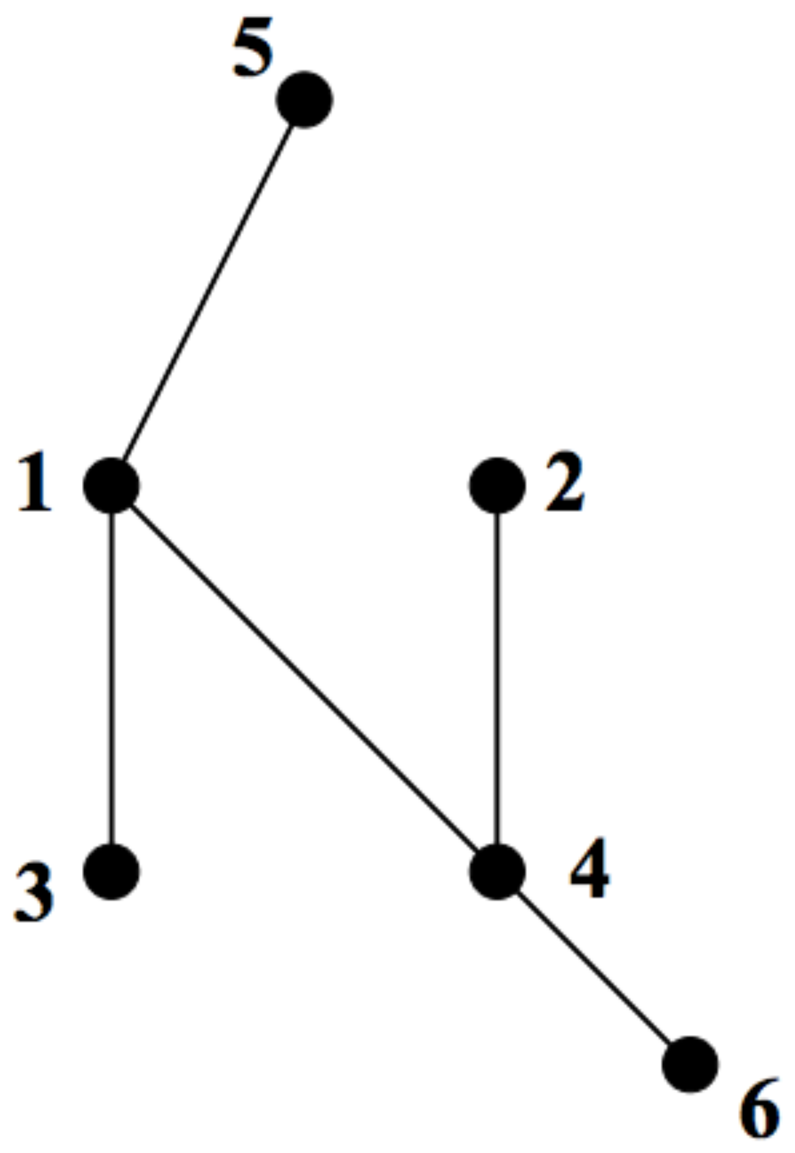
Enter answers on formatted sheet, front of exam book.
Show enough to get partial credit if answer is incorrect.
Can be turned in entirely on paper, there's also the option of uploading anything electronic to the course website during exam.

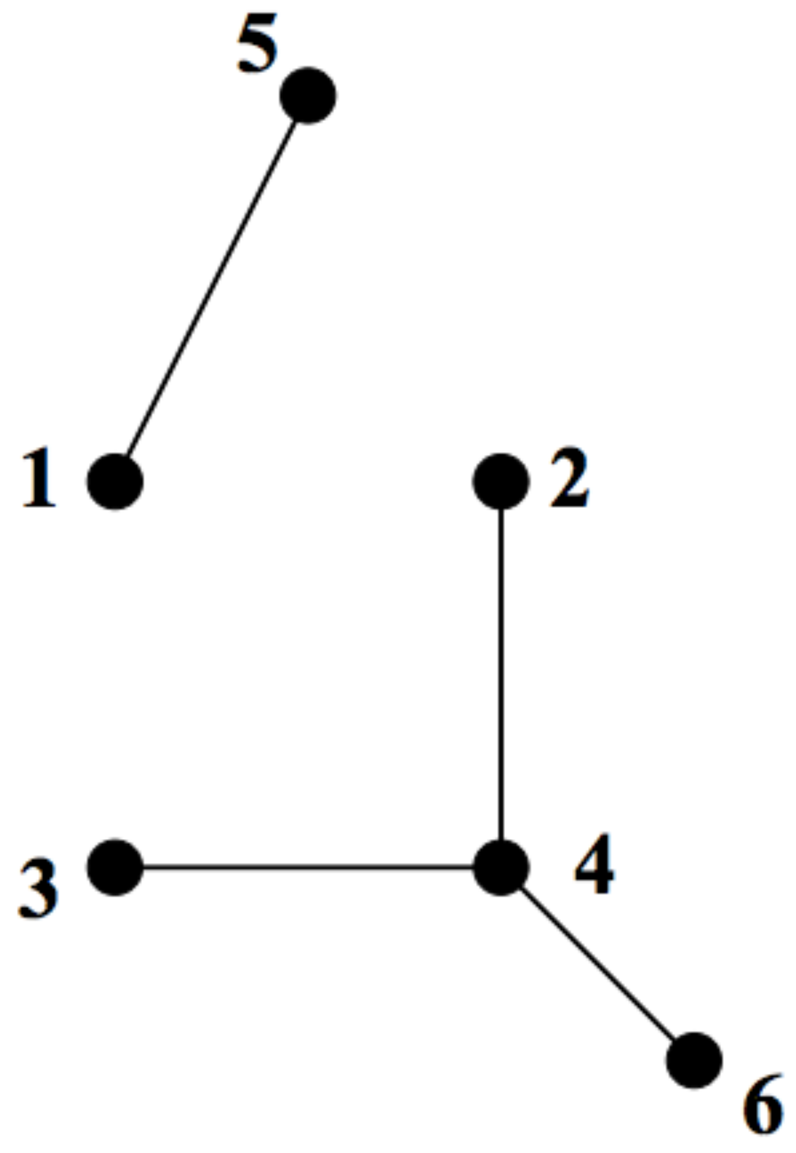
A tree is a connected graph with no cycles.

A forest is a graph with each connected component a tree.

A leaf in a tree is any vertex of degree 1.

Example: a tree and a forest of 2 trees





Proposition. For any tree $T = (V, E)$ with $|V| = n$, $|E| = n - 1$.

Consider any leaf of T
This vertex is adjacent to exactly one edge.

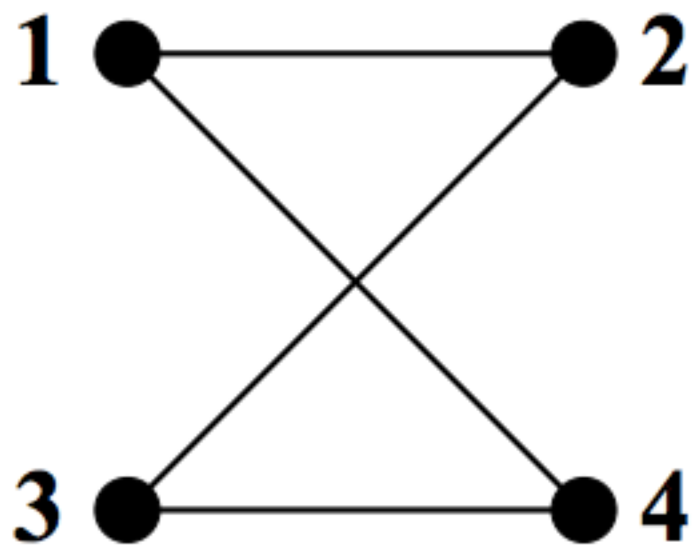
Remove this vertex and edge, which contributed 1 each to the number of vertices $|V|$ and number of edges $|E|$, leaves $|E| - |V|$ unchanged.

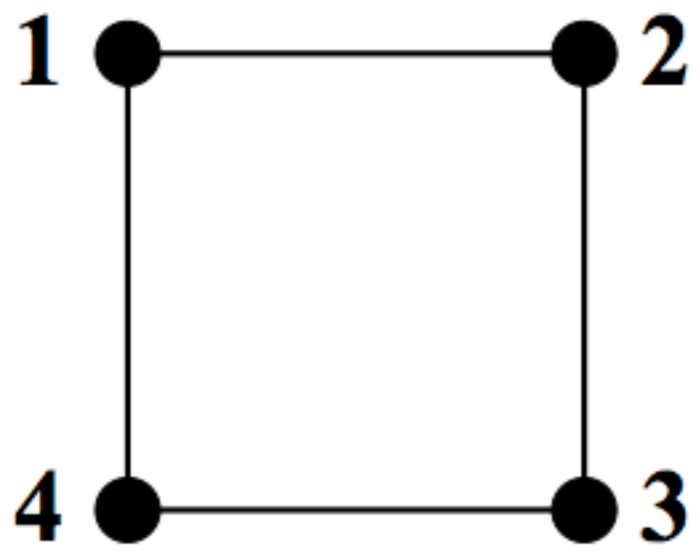
Continue removing leaf / edge pairs until left with just a single edge.

A graph with a single edge has one more vertex than edge, hence the total number of edges is one less than the total number of vertices.

A graph G is planar if there exists an embedding of G into the plane such that no two edges cross.

Example: The graph on 4 vertices with edges (1, 2) (2, 3) (3, 4) and (4, 1) is planar, because it can be drawn with no edges crossing (though it can also be drawn with 1 edge crossing …)





The edges of a planar embedding of a graph divide the plane into regions. Let f be the number of regions of a planar graph, e the number of edges and v the number of vertices.

Theorem (Euler's formula): For any connected planar graph, $v - e + f = 2$.

Proof. We proceed by induction on the number of edges e.

$e = 1$. There is only one such graph. This graph has $v = 2$, $e = 1$ and $f = 1$

Consider a connected planar graph G with $n + 1$ edges, v vertices and f regions Form G' with statistics e', v', and f' by removing any edge which still leaves another connected graph. Then $v' = v$, $e' = e - 1$, and $f' = f - 1$

Therefore we have $2 = v' - e' + f' = v - (e - 1) + (f - 1) = v - e + f$.

Therefore we have $2 = v' - e' + f' = v - (e - 1) + (f - 1) = v - e + f$.

Except: In forming G', it could have been that removing any edge of G resulted in a disconnected graph.

In this case, G is a tree. (why?)

But using the earlier proposition, we know that for any tree

$$v - e + f = v - (v - 1) + 1 = 2.$$

Some graph combinatorics:

total degree = 2|E|

mean degree = 2|E|/|V|

random graph:

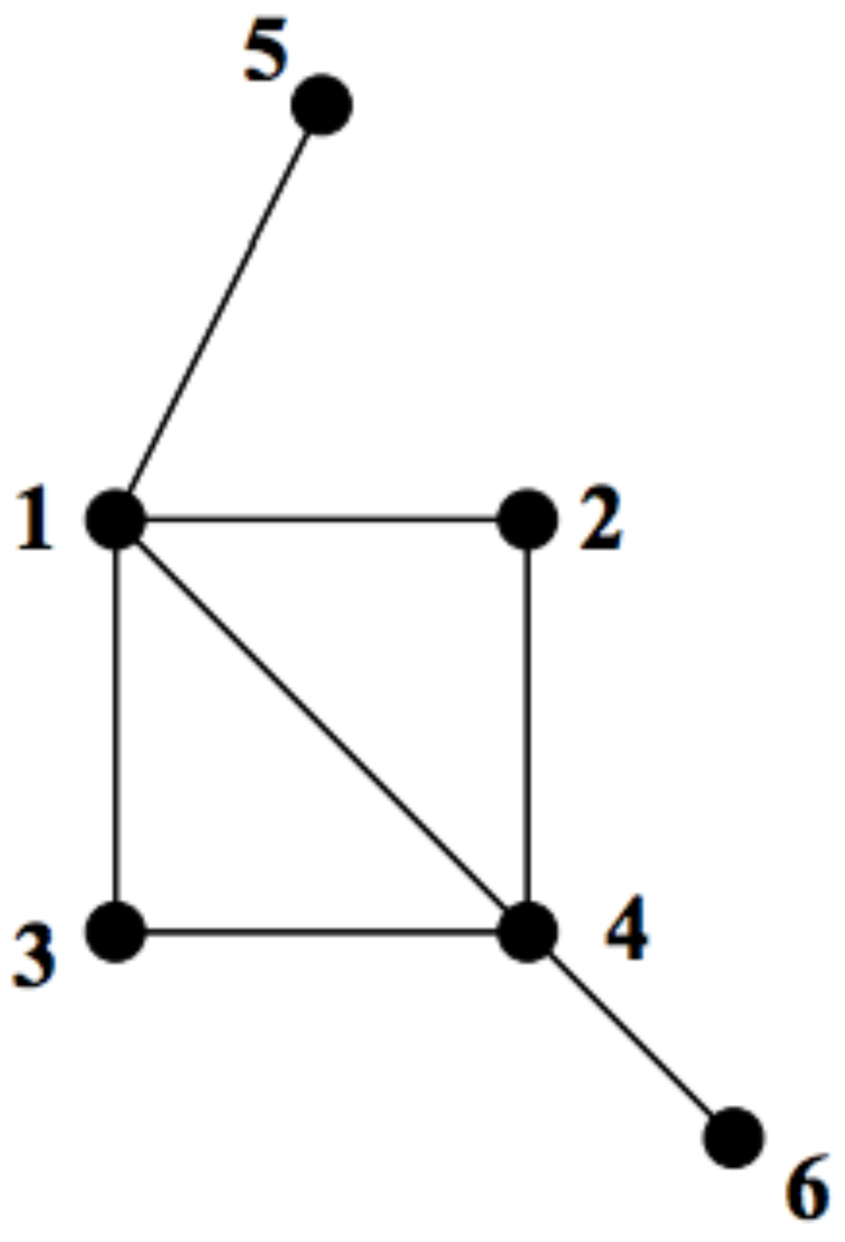
expected mean degree=

probability of degree m =

The length $l(p)$ of a path p is the number of edges in the path.
The distance between two vertices v_1 and v_2, written $d(v_1, v_2)$,
is the length of the shortest path connecting the vertices:

$$d(v_1, v_2) = \min\{\, l(p) \mid p \text{ is a path connecting } v_1 \text{ and } v_2\}$$

The distances of all pairs of vertices in a graph can be represented as a matrix.

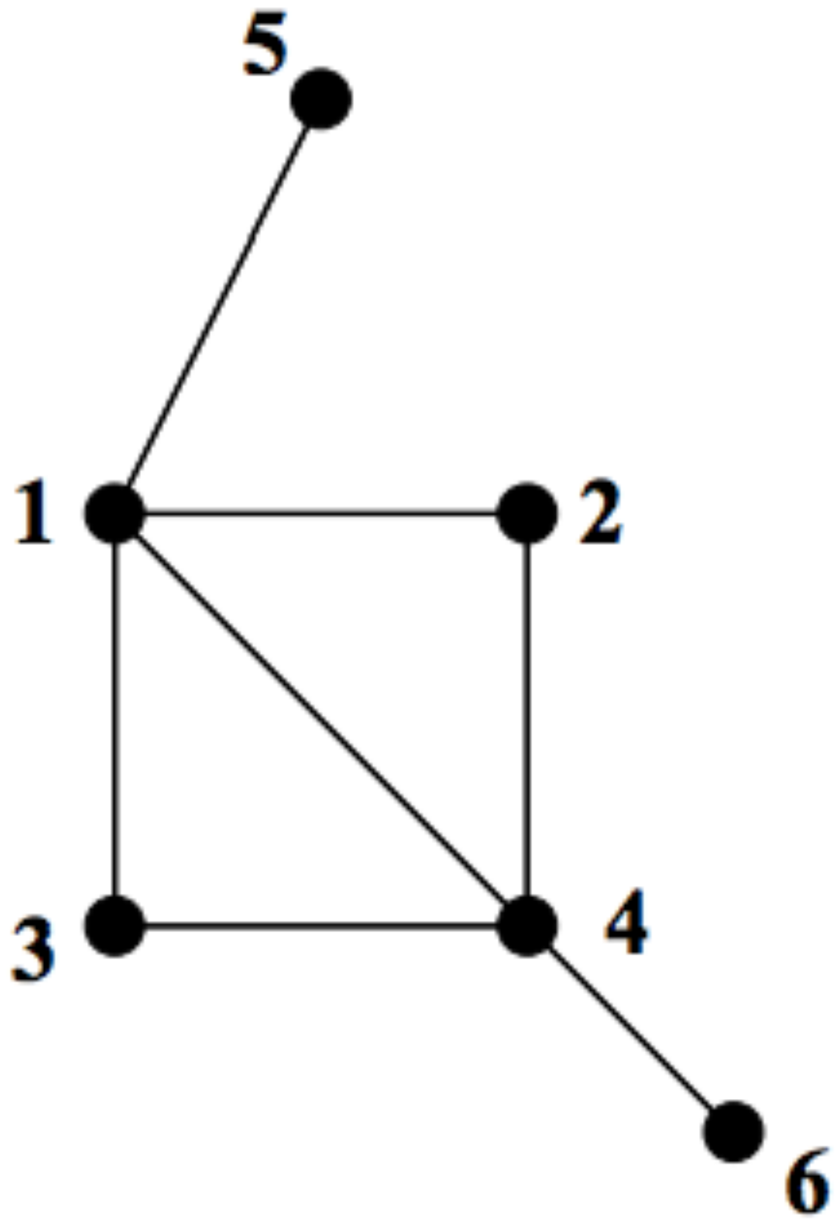


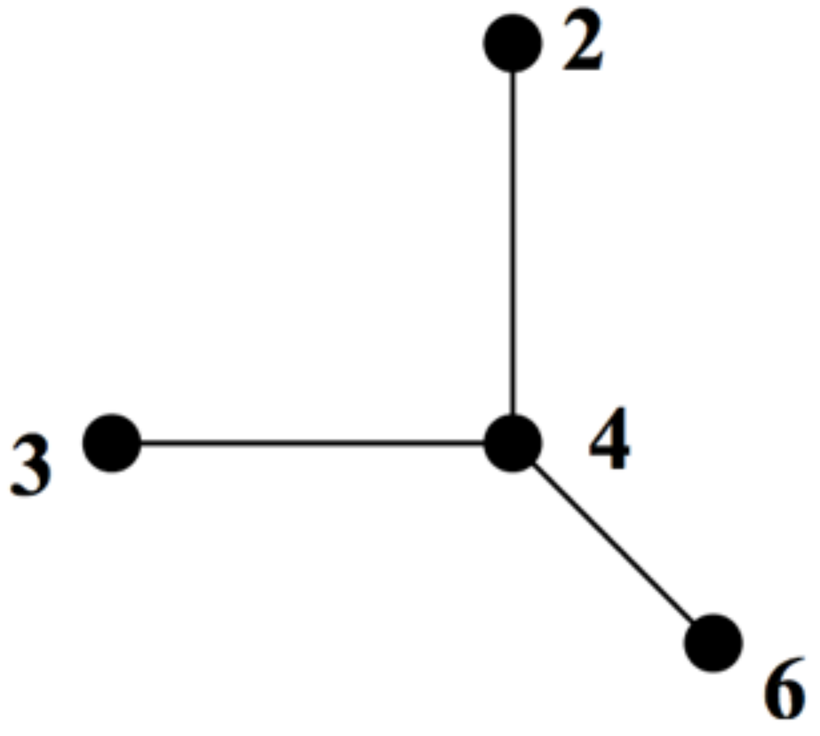
$$\mathbf{D} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 & 2 & 2 \\ 1 & 2 & 0 & 1 & 2 & 2 \\ 1 & 1 & 1 & 0 & 2 & 1 \\ 1 & 2 & 2 & 2 & 0 & 3 \\ 2 & 2 & 2 & 1 & 3 & 0 \end{pmatrix}$$

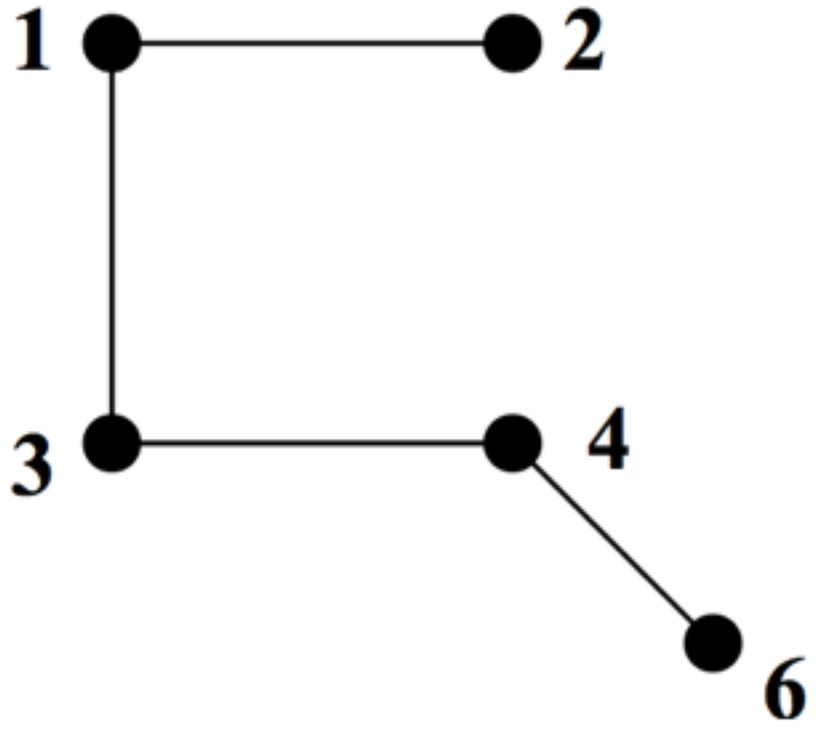
The diameter of a graph G is the maximum distance between any two vertices in G:

$$\mathrm{diam}(G) = \max \{ d(v_1, v_2) \mid v_1, v_2 \text{ are vertices of } G \}$$

Example: The diameter of G_1 is 3. The diameter of the first subgraph in from earlier is 2 and the diameter of the second subgraph is 4.

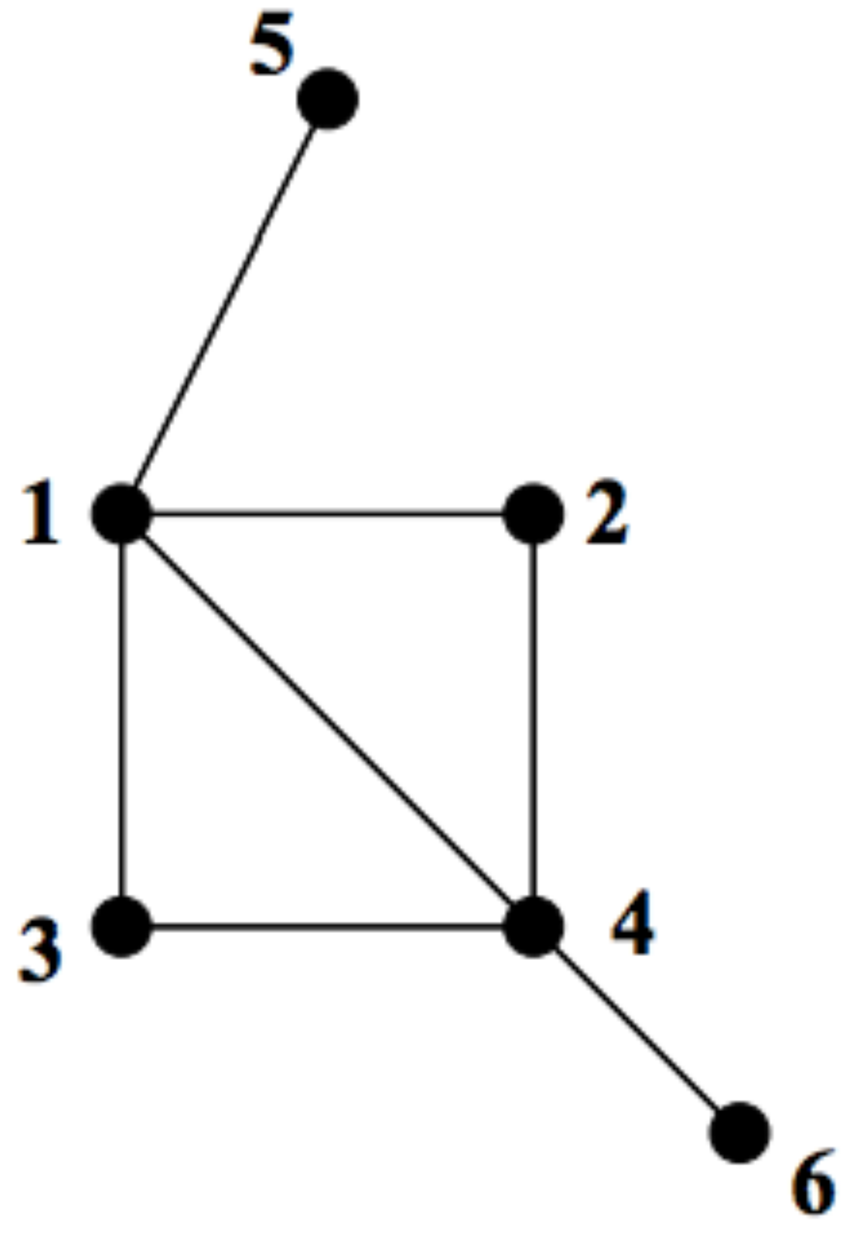






Another matrix associated to a graph G is the adjacency matrix which has entry ij = 1 if $(v_i, v_j) \in E(G)$, and equal to 0 otherwise.

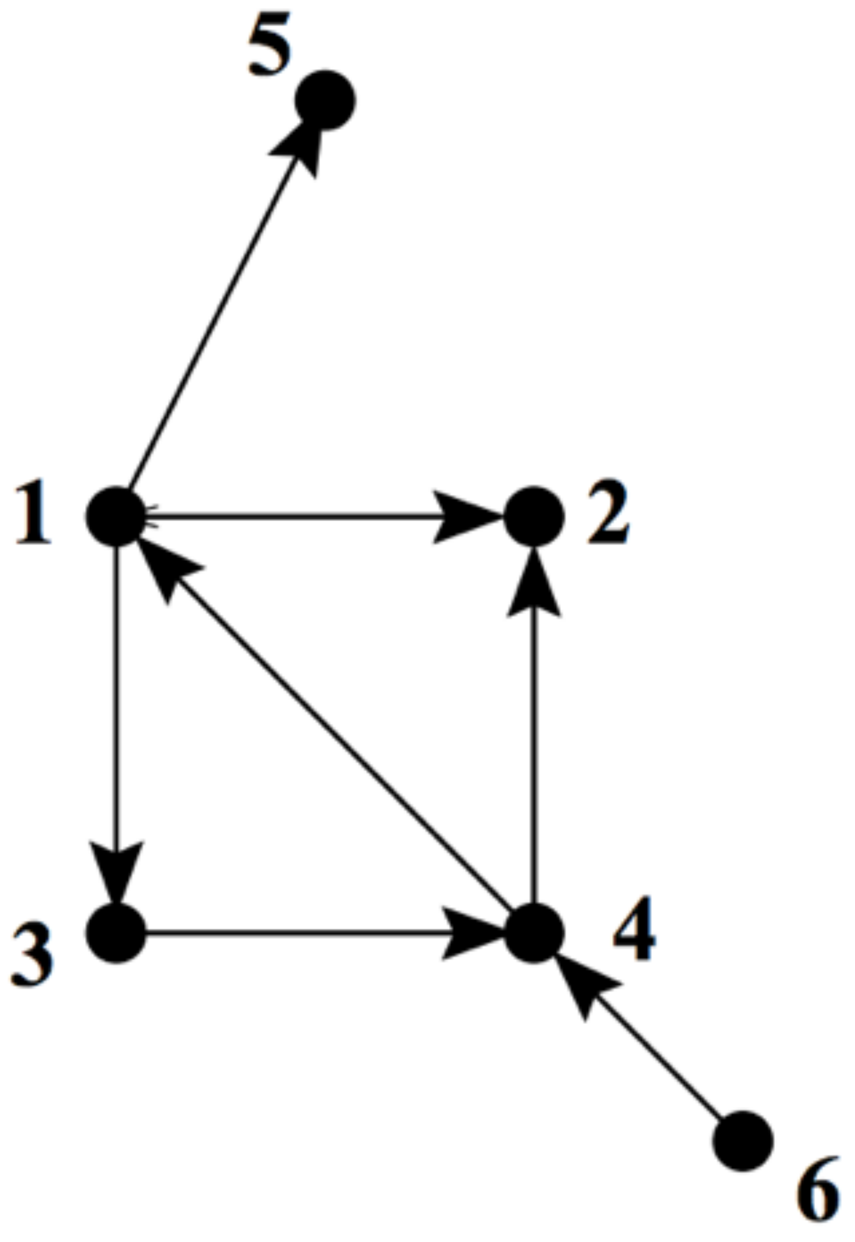
Example: adjacency matrix for G_1:



$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Another matrix associated to a graph G is the adjacency matrix which has entry ij = 1 if $(v_i, v_j) \in E(G)$, and equal to 0 otherwise.

Example: adjacency matrix for directed version of G_1:



$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

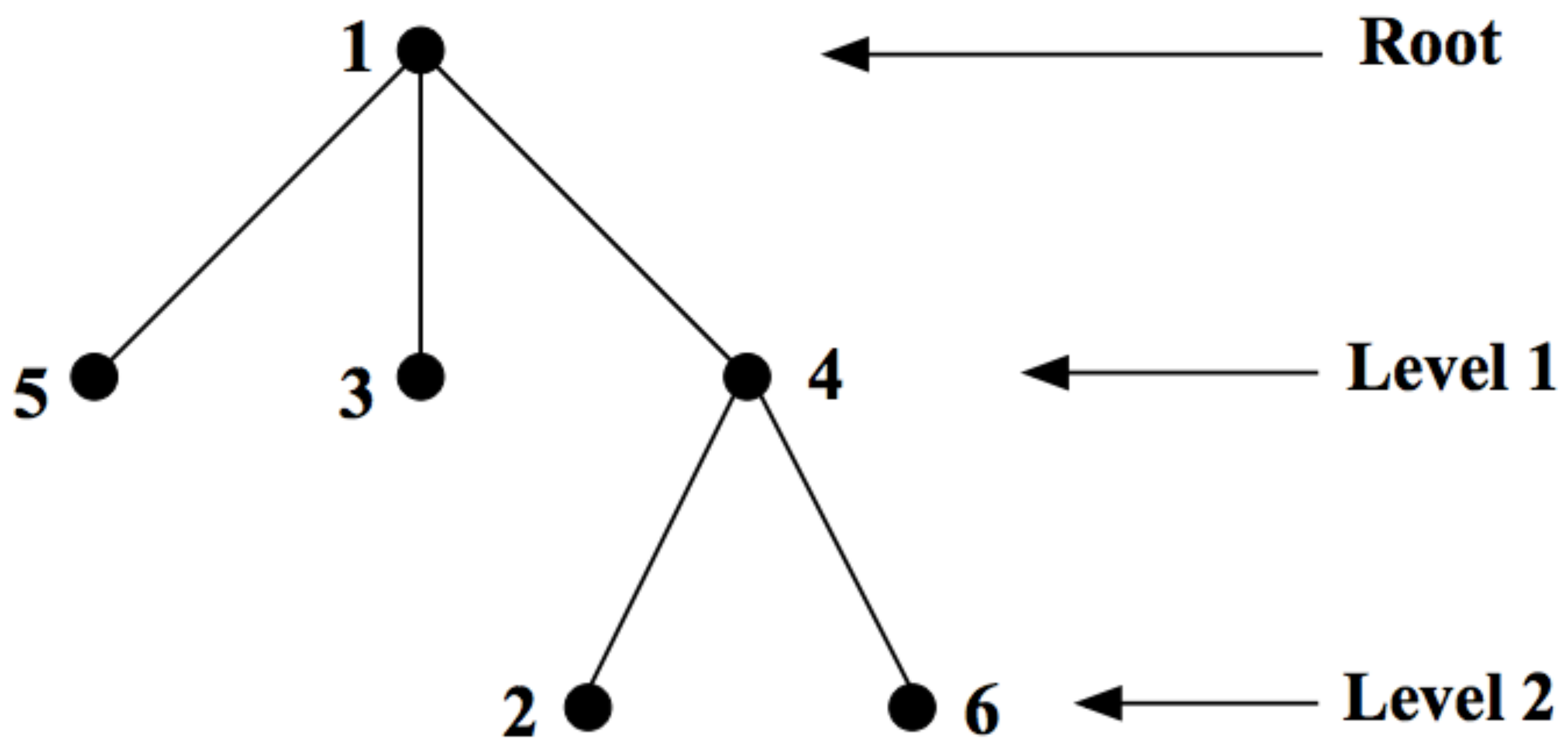
A rooted tree is a graph with a distinguished vertex called the root of the tree.

A rooted tree is frequently drawn with the root on the top and the other vertices in layers below, corresponding to their distance from the root.

The collection of vertices at distance i from the root is called the i'th level of the tree.

The depth of a rooted tree is the number of levels of the tree.

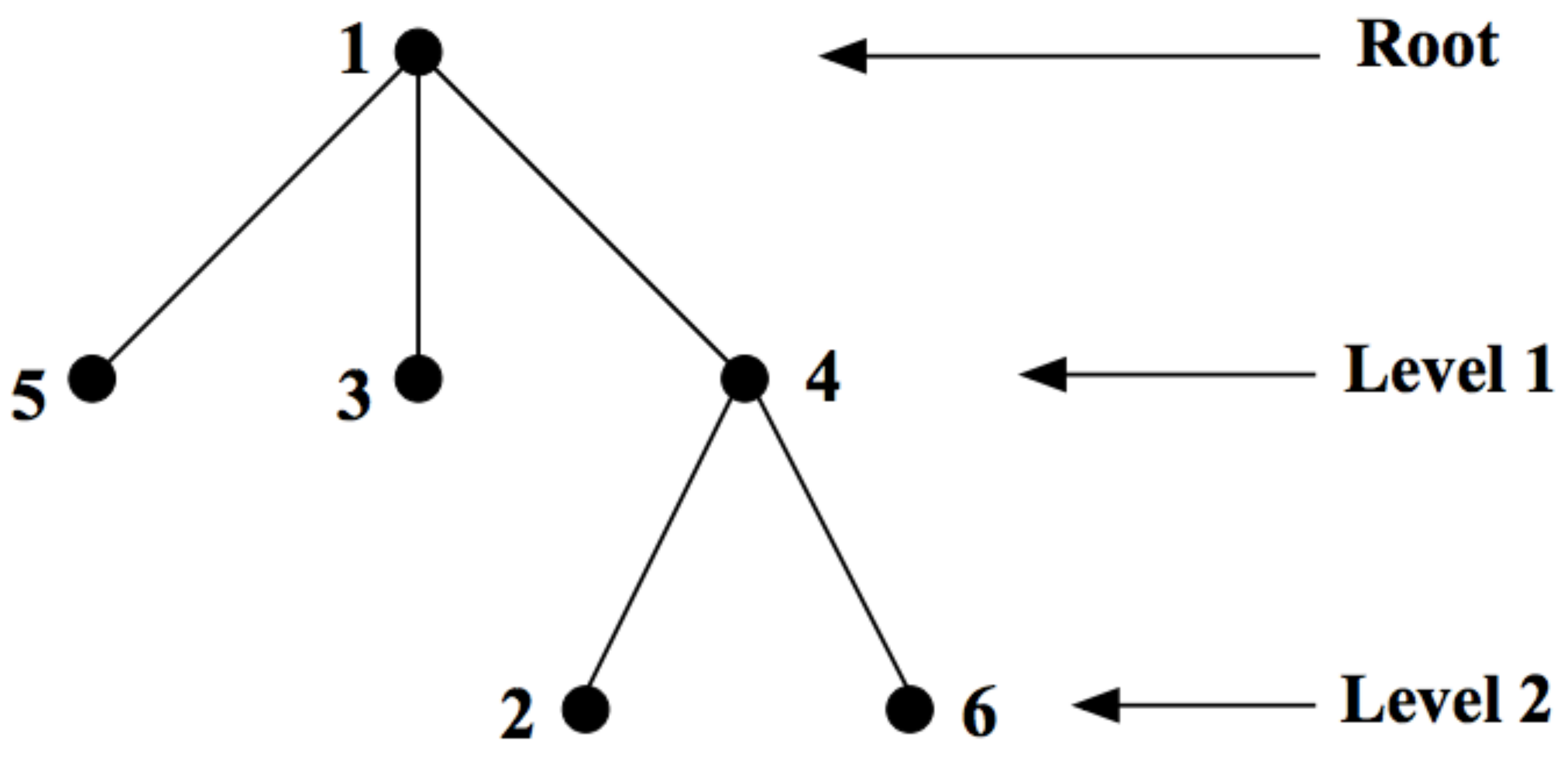
Example:



For a vertex of a rooted tree at level i,
its neighbor on level i – 1 is called its parent
and its neighbors on level i + 1 are called its children.

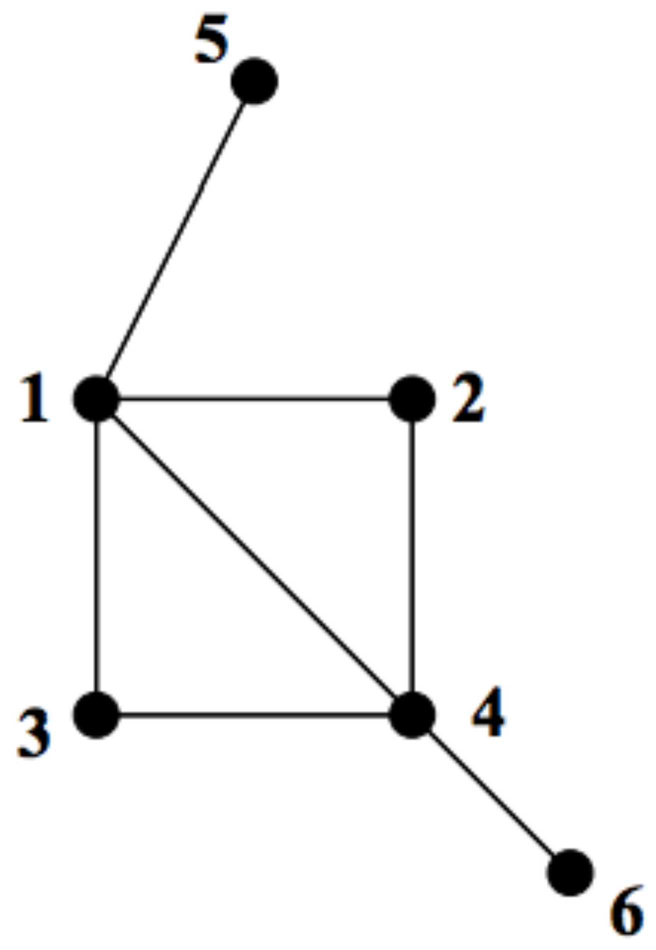
A rooted binary tree is a rooted tree such that each vertex has 2 children (except for leaf nodes)

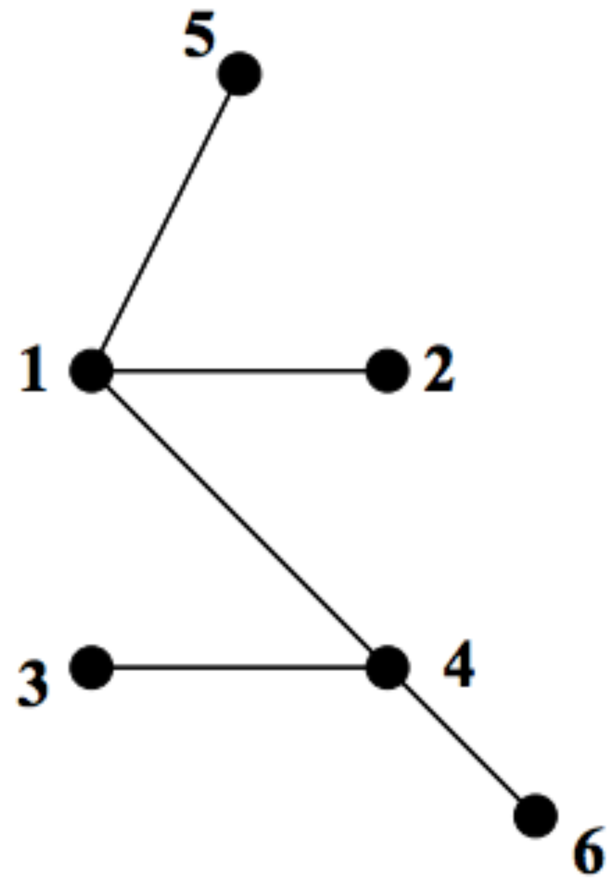
Example: vertex 1 has children {3, 4, 5} and vertex 4 has children {2, 6}.
This tree is not a binary tree because vertex 1 has three children.
If we removed vertex 5, the resulting tree would be binary.

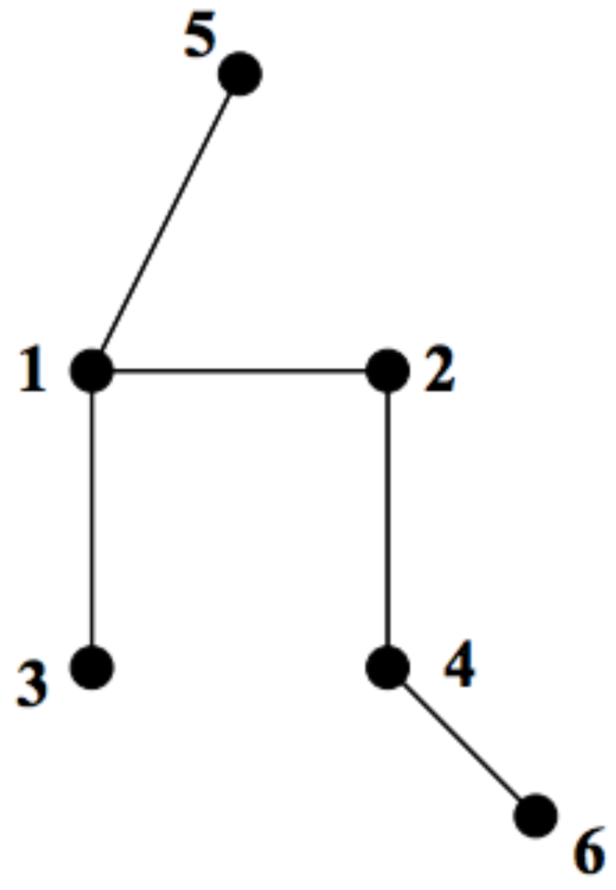


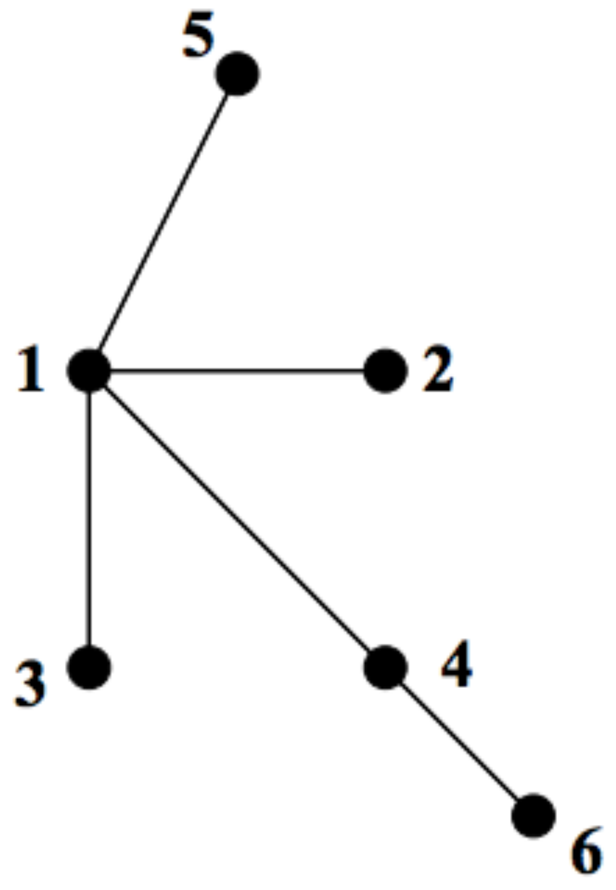
A spanning tree $T = (V', E')$ of a graph $G = (V, E)$ is a subgraph such that T is a tree and $V' = V$.

Example: three different spanning trees of G_1









Next we consider ways of finding spanning trees of arbitrary simple graphs. Here we consider two algorithms, breadth first search and depth first search.

Breadth first search (BFS) and Depth First Search (DFS)

Must have an ordering on the vertices of the graph.

In most examples here, the vertices have been labeled by $\{1, 2, \ldots, n\}$
where n is the number of vertices.
Gives a natural ordering

These algorithms output a rooted spanning tree.

DFS algorithm

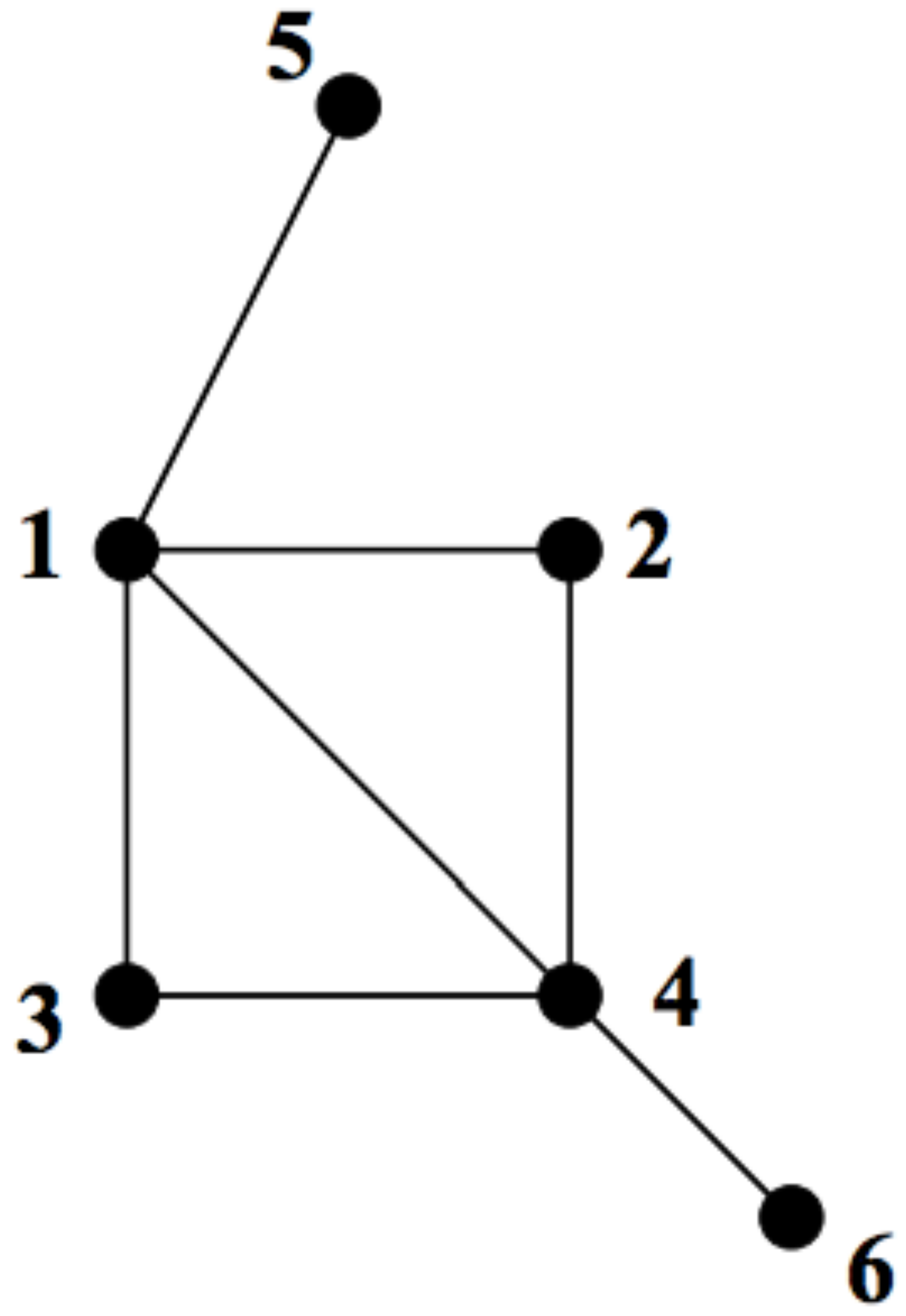
Initialize $T = (V, E)$:
$V = v_1$
$E = \{\}$

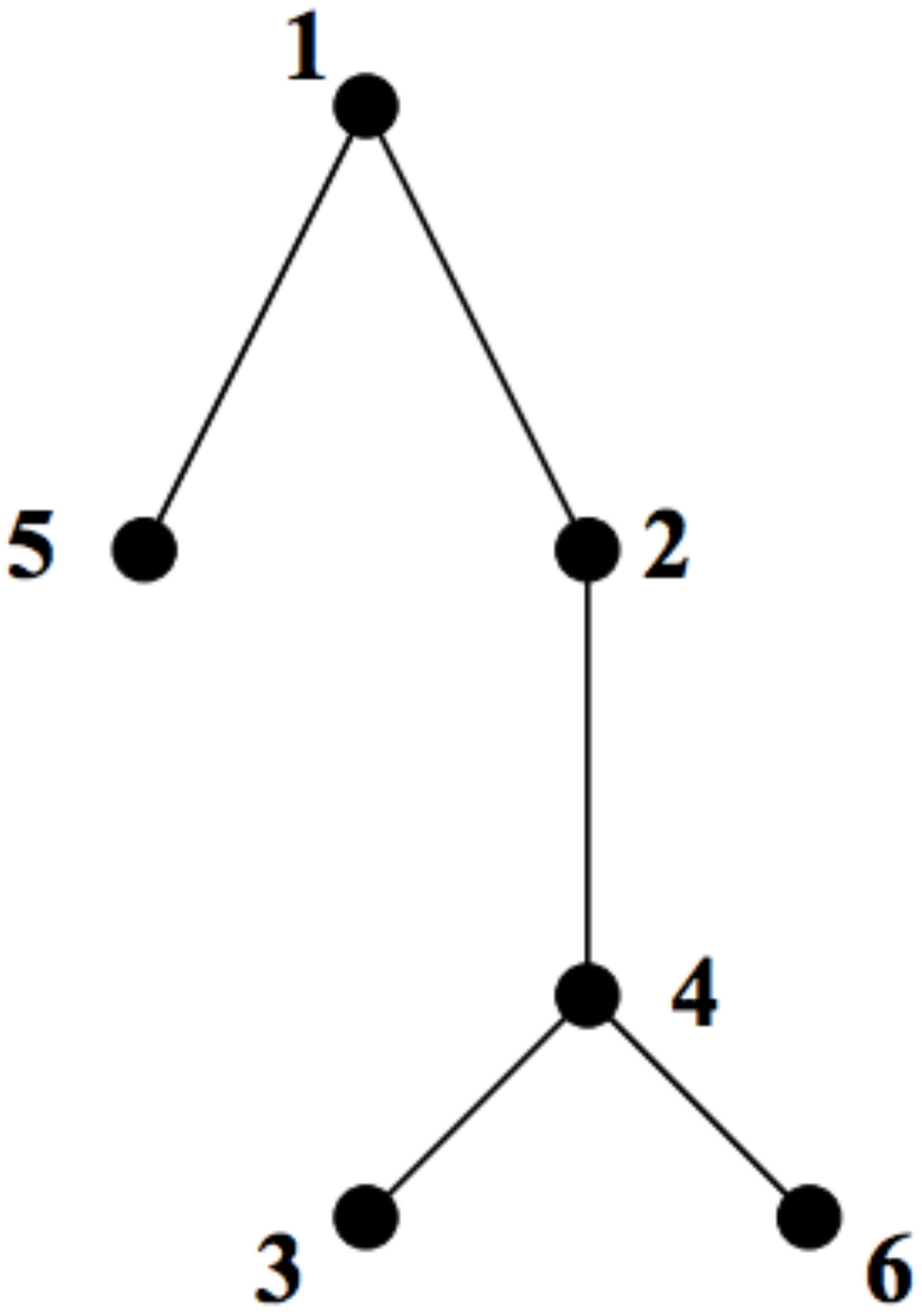
$v = v_1$

---

if there's an edge (v,w) such that w is not already in V:
    add the edge to E
    $v = w$ (i.e., make w the new v), repeat
else:
    if $v == v_1$: done (have made it back up to root)
    else: v = parent of v (make parent of v the new v), and repeat

DFS





if there's an edge (v,w) such that w is not already in V:
    add the edge to E
    v = w (i.e., make w the new v), repeat
else:
    if v == v_1 : done (have made it back up to root)
    else: v = parent of v (make parent of v the new v), and repeat

BFS algorithm

Suppose original G has n vertices

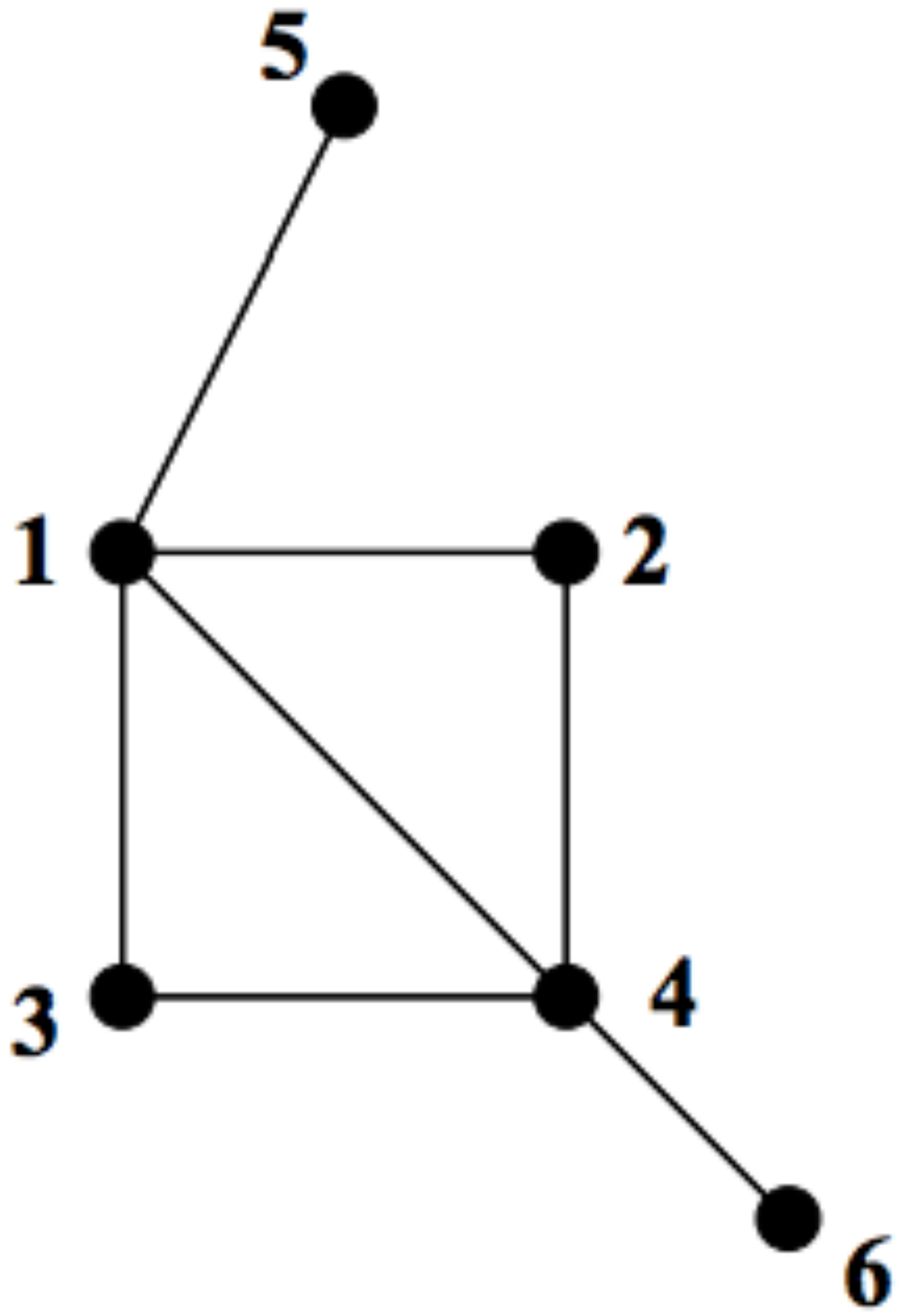
Initialize $T = (V, E)$:
$V = \{v_1\}$
$E = \{\}$

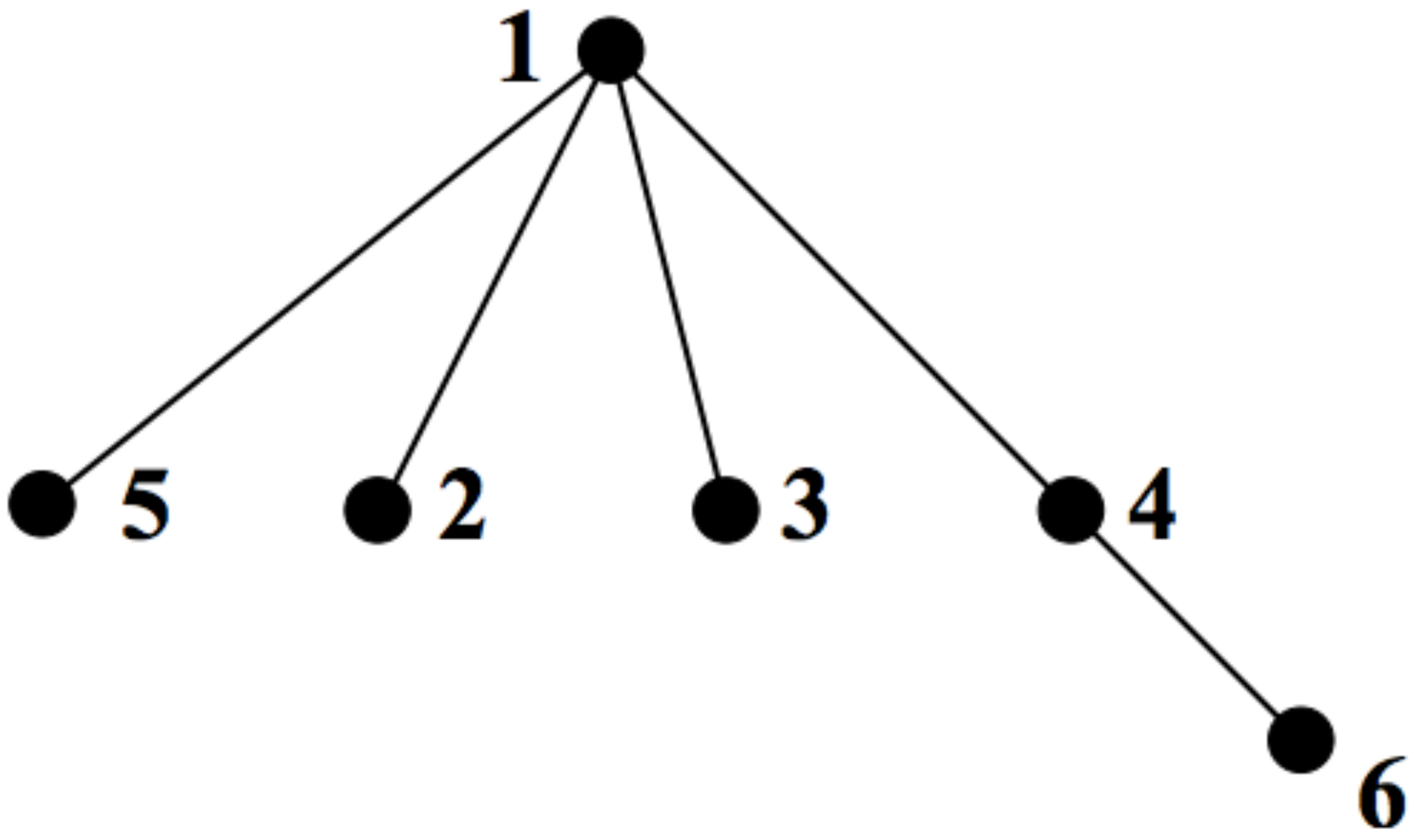
$v = v_1$

---

for all neighbors w of v not in V:
    add w to V
    add edges (v,w) to E
if $|E| == n - 1$: stop
else: v = next element of V, and repeat

BFS





for all neighbors w of v not in V:

    add w to V

    add edges (v,w) to E

if |E| = n - 1: stop

else: v = next element of V, and repeat