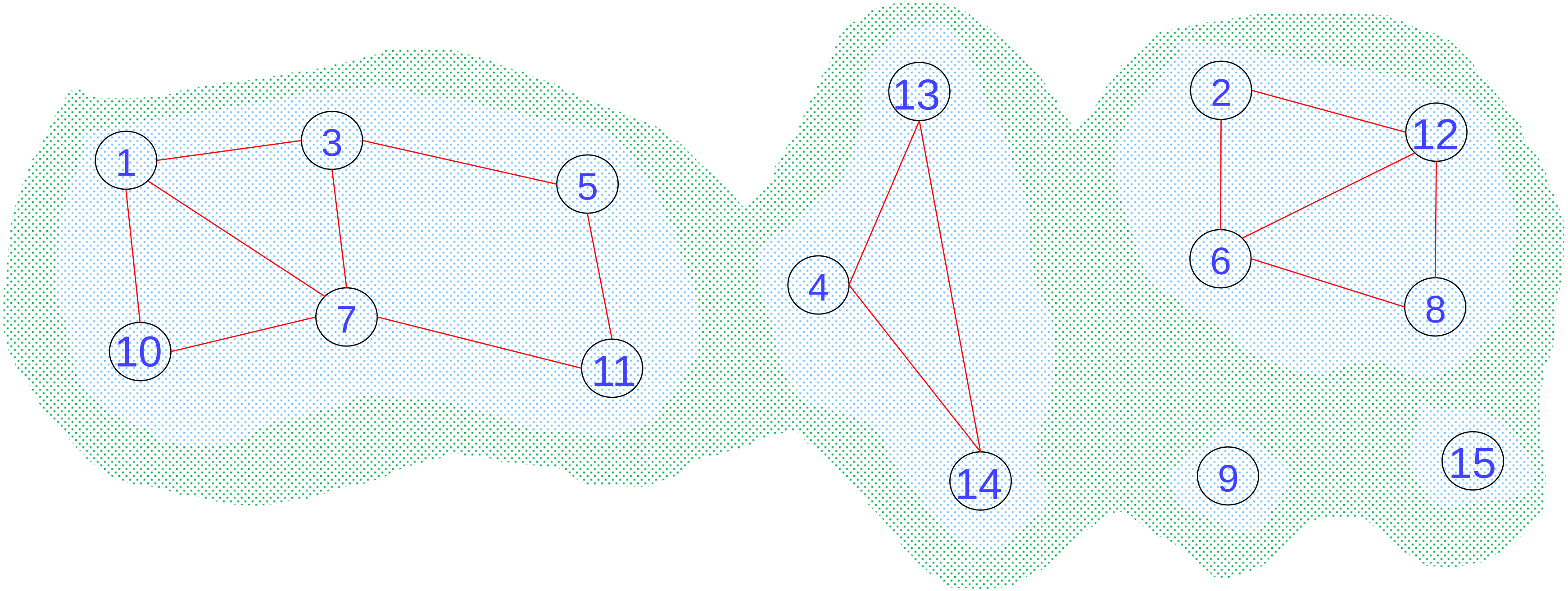
Applications of Breadth-First Search

Application: Connected Components using BFS

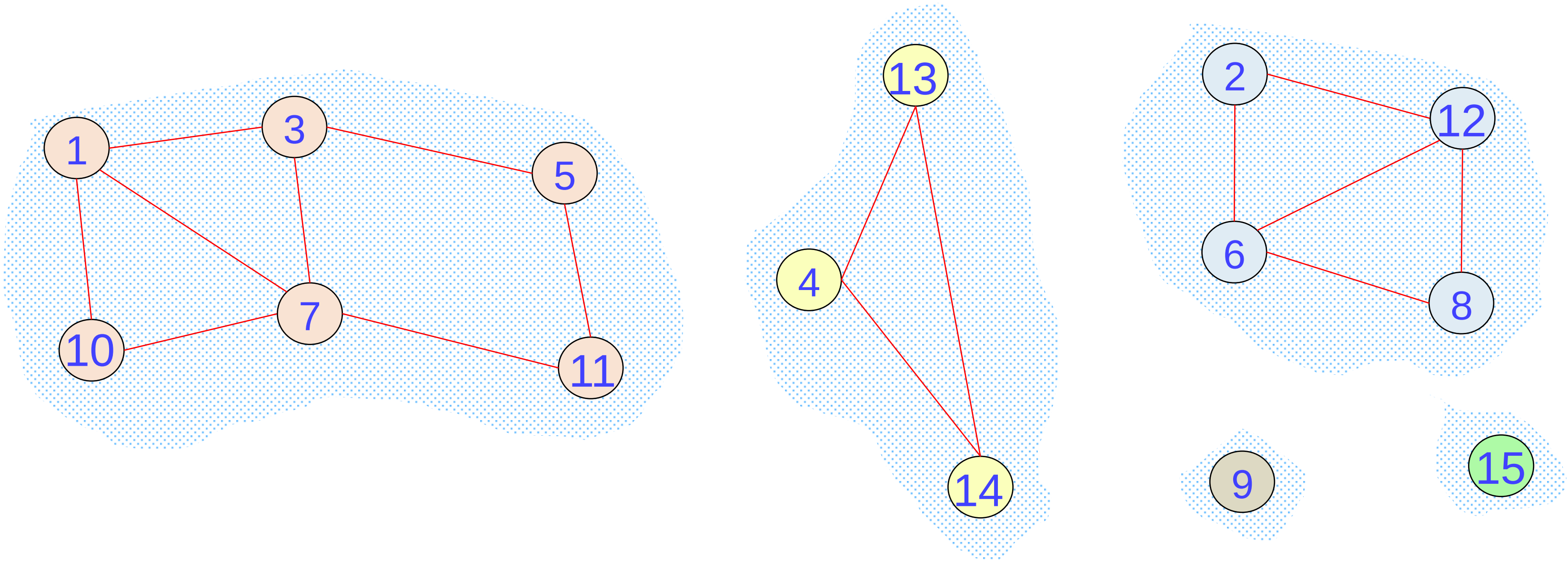
- How many connected components does this graph ***G*** = ***(V, E)*** have?



- To find one procedure which will label the vertices in terms of their components

Application: Connected Components using BFS

- How many connected components does this graph ***G*** = ***(V, E)*** have?



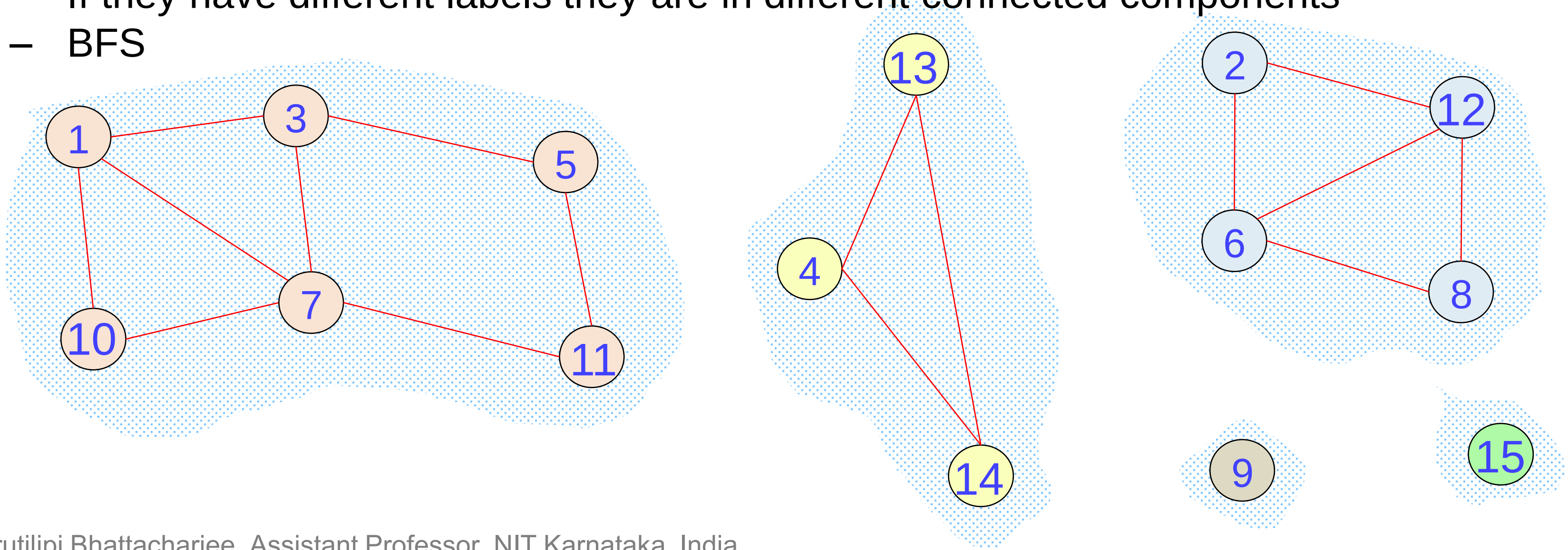
- To find one procedure which will label the vertices in terms of their components

Application: Connected Components using BFS

- Let us have an array, ComponentNumber =

1	3	1	2	1	3	1	3	4	1	1	3	2	2	5

- Given any two vertices, in constant time can we determine if they are in the same connected component?
  - If they have the same label then they are in the same connected component
  - If they have different labels they are in different connected components
  - BFS

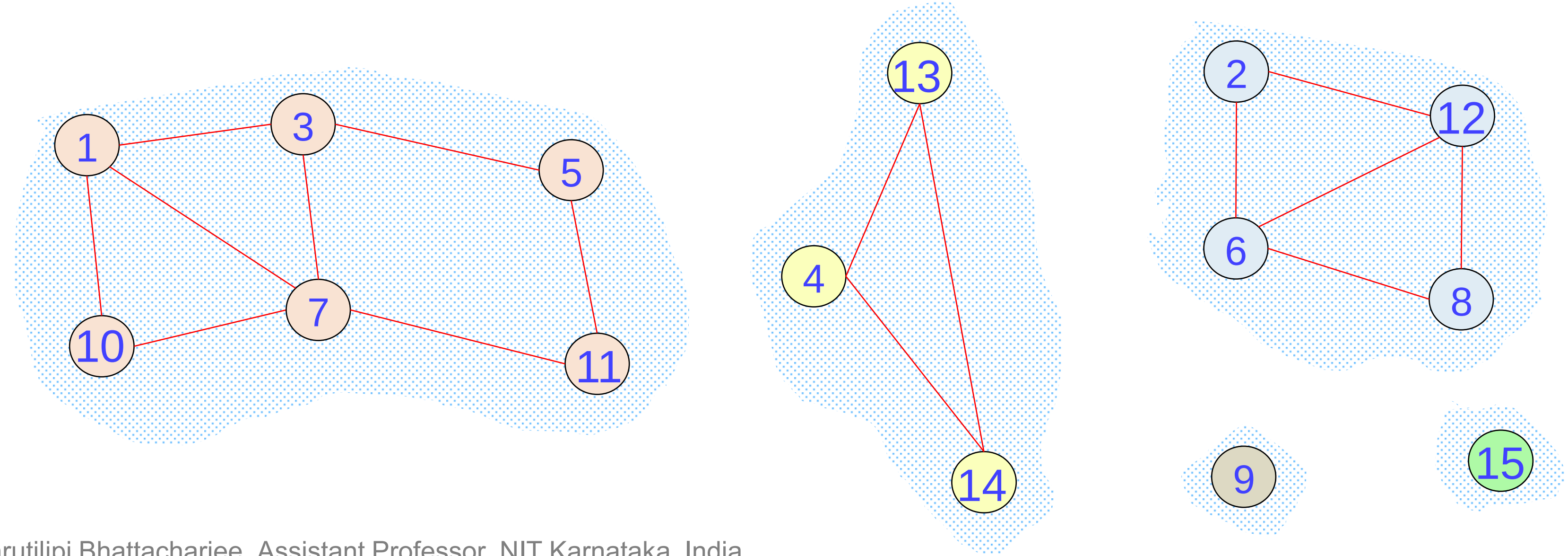


Application: Connected Components using BFS

- Take a random a starting vertex and do a BFS from there
- All the vertices in the same component would get visited
- ComponentNumber =

1	0	1	0	1	0	1	0	0	1	1	0	0	0	0

- While visiting these vertices, we will assign them a ComponentNumber = **1**
- Now pick any vertex with ComponentNumber = 0

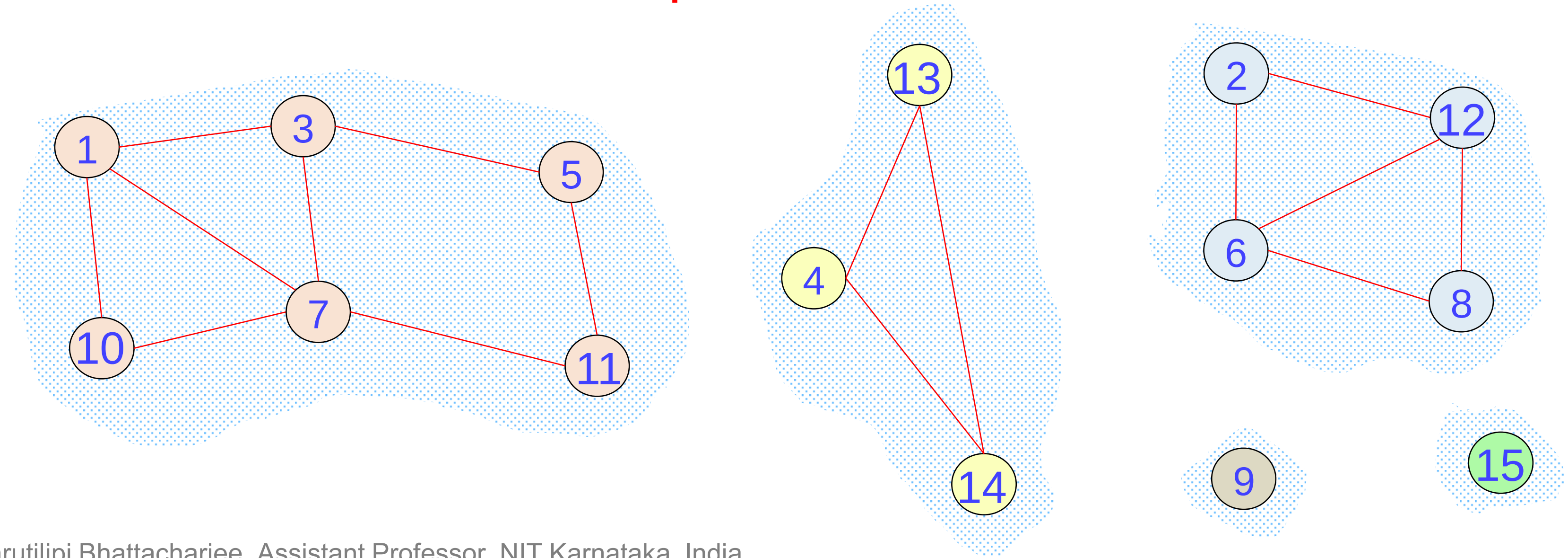


Application: Connected Components using BFS

- After second BFS, all the vertices in the second component will be assigned the ComponentNumber = 2
- ComponentNumber =

1	2	1	0	1	2	1	2	0	1	1	2	0	0	0

- Repeat this process
- How many times the ComponentNumber[] array will be processed?



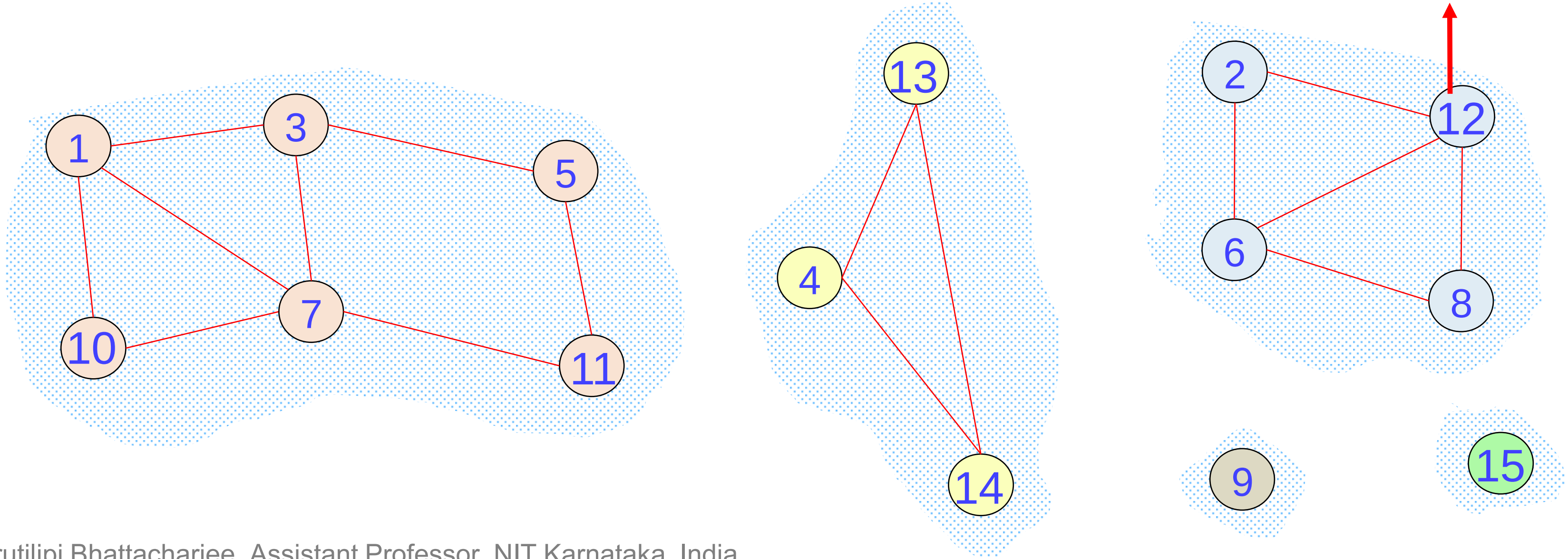
Application: Connected Components using BFS

- After third BFS
- ComponentNumber =

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

- After fourth BFS
- ComponentNumber =

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

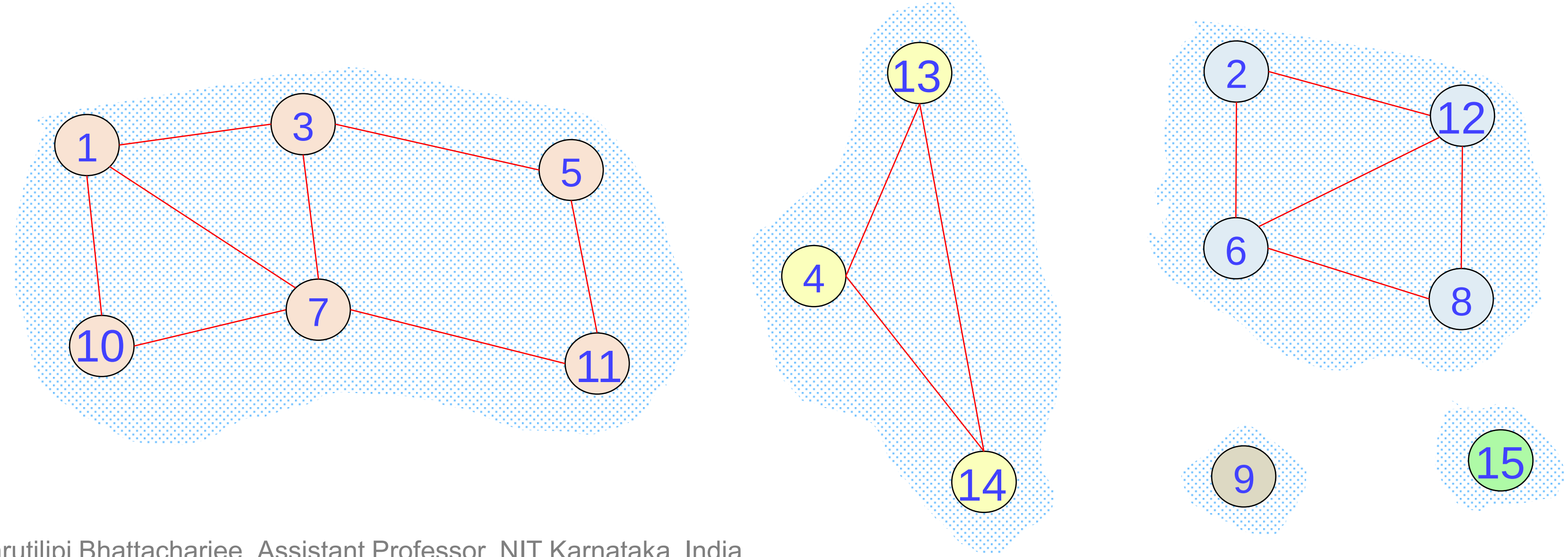


Application: Connected Components using BFS

- After fifth BFS
- ComponentNumber =

1	2	1	3	1	2	1	2	4	1	1	2	3	3	5

- Total running time: ***O(V + E)*** [not a connected component, can be ***O(max(V, E))***]



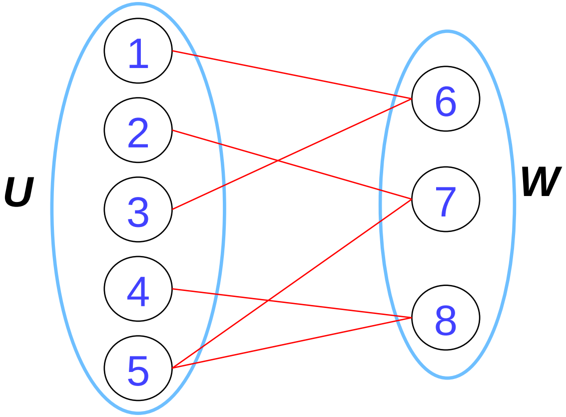
Application: Bipartite Graph using BFS

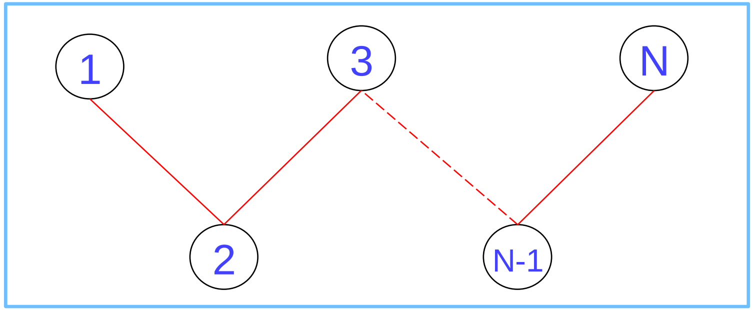
- ***G*** = ***(V, E)*** is a ***bipartite graph*** if there exist a partition of ***V*** into two disjoint and independent sets ***U*** and ***W*** partitions such that every edge has one end point in ***U*** and the other in ***W***, where

$$\boldsymbol{U} \cup \boldsymbol{W} = \boldsymbol{V}$$

$$\boldsymbol{U} \cap \boldsymbol{W} = \boldsymbol{\Phi}$$

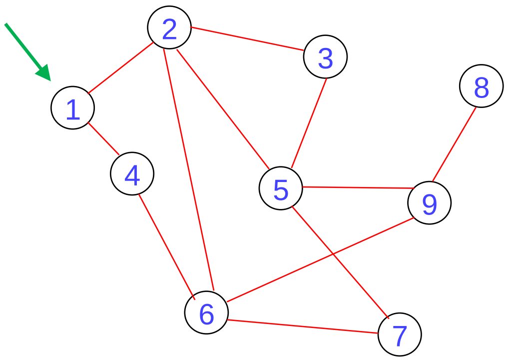
- Equivalently, a ***bipartite graph*** is a graph that does not contain any odd-length cycles

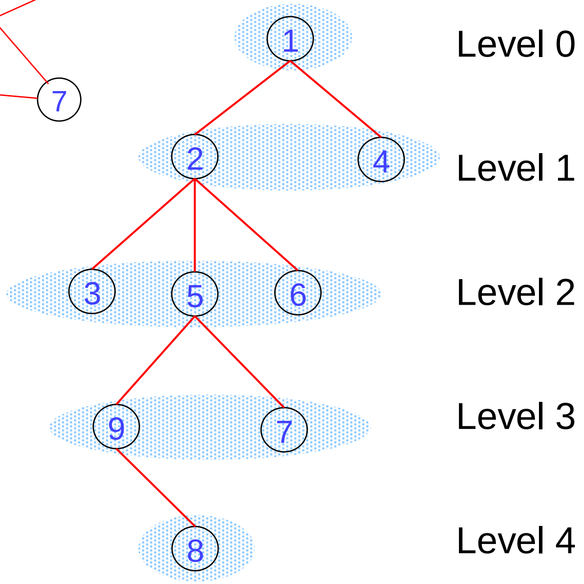




Application: Bipartite Graph using BFS

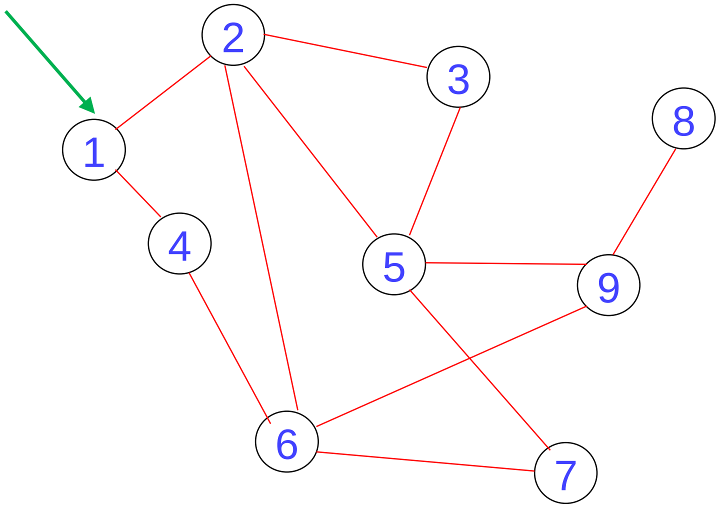
- Given a graph ***G*** = ***(V, E)***, how will you check if it is a ***bipartite graph***?
  - Let's assume ***G*** is connected
  - Otherwise we have to check it for each connected component
    - If each connected component is bipartite, then the graph is bipartite
    - If some one connected components is non-bipartite then the graph is not bipartite
  - Start a BFS from any vertex
  - Recall that BFS divide the graph up into layers into levels in BFS tree

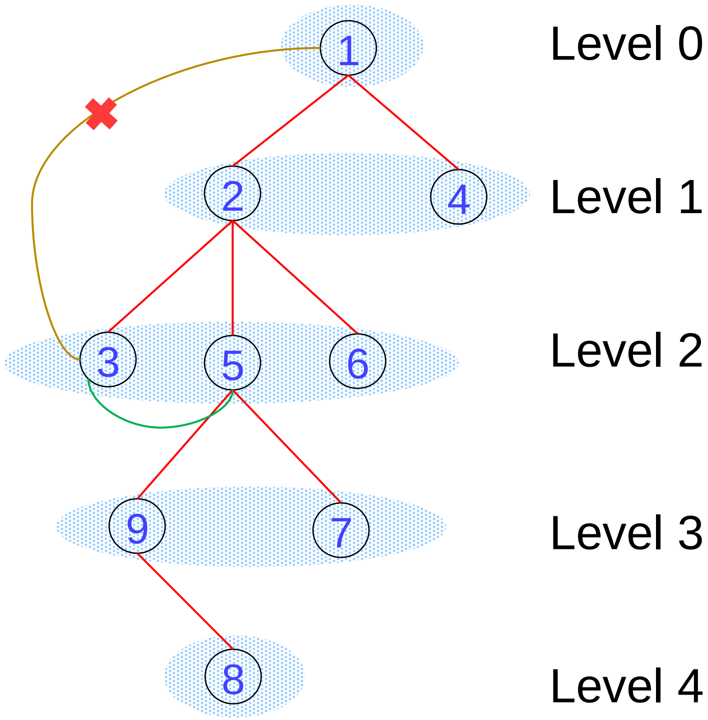




Application: Bipartite Graph using BFS

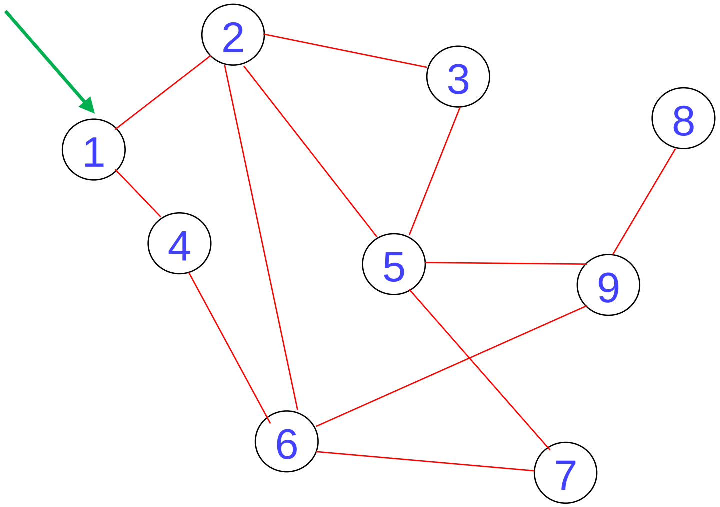
- The BFS tree will organize the vertices in the levels such that all edges connects between adjacent levels or within a level
- There would be no edges in ***G*** which jump levels

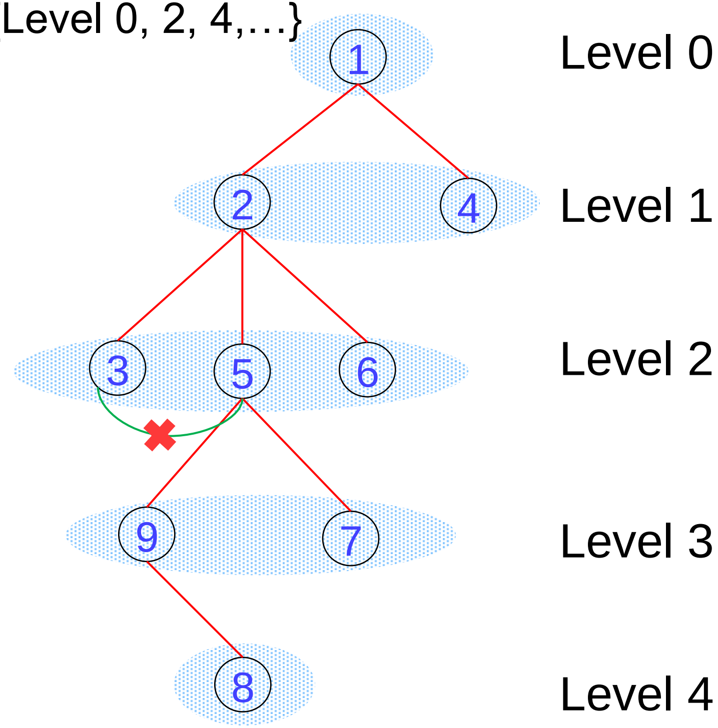




Application: Bipartite Graph using BFS

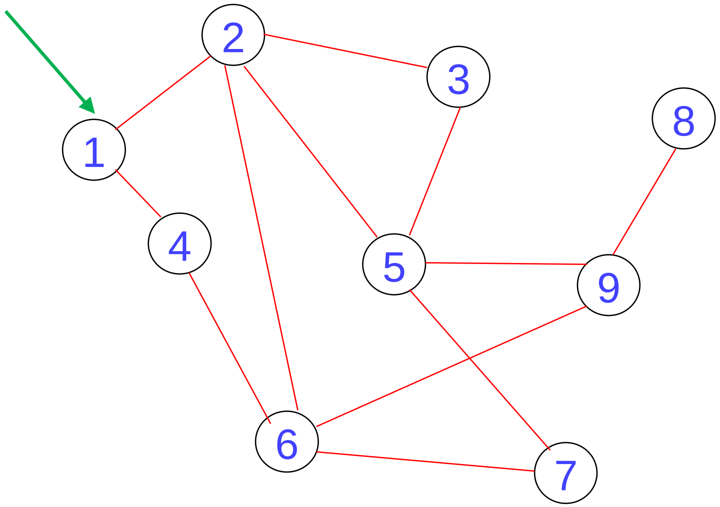
- If all edges connects between adjacent levels, is the graph ***bipartite***?
- How?
  - ***U*** = Vertices{Level 1, 3,…} and ***W*** = Vertices{Level 0, 2, 4,…}
  - The ***G*** is ***bipartite***

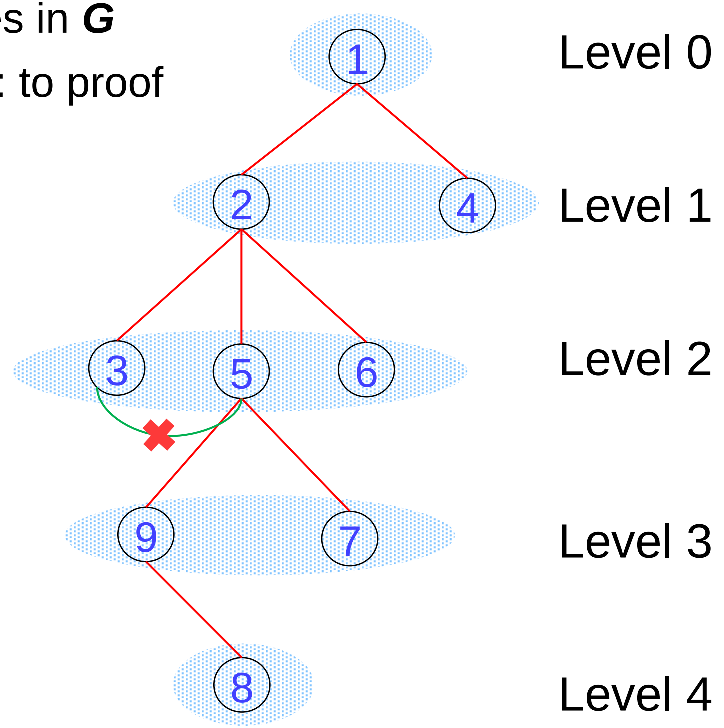




Application: Bipartite Graph using BFS

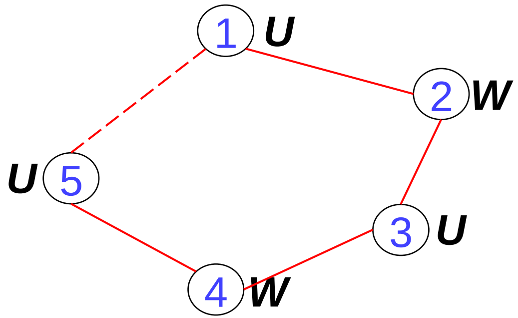
- If an edge connects within the same level, is the graph ***bipartite***?
  - The ***G*** is not ***bipartite***
  - Also these same level edges forms an odd cycles in ***G***
  - If ***G*** has an odd cycle then ***G*** cannot be bipartite: to proof





Application: Bipartite Graph using BFS

- Proof: To prove by contradiction

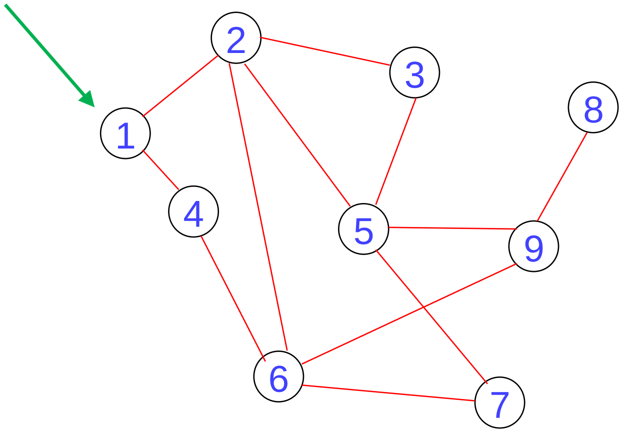


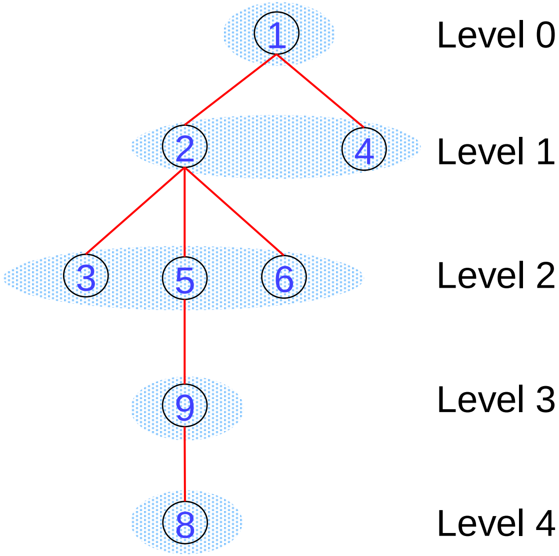
- If all the cycles in a graph ***G*** are even in length, then, does that mean that the graph is **bipartite**?
  - If ***G*** has an odd cycle then ***G*** cannot be bipartite
  - There would be no edges in ***G*** which jump levels
  - We can marks alternate level vertices in the similar group ***U*** and ***W***

- Time complexity: Same as BFS!

Application: Shortest Path using BFS

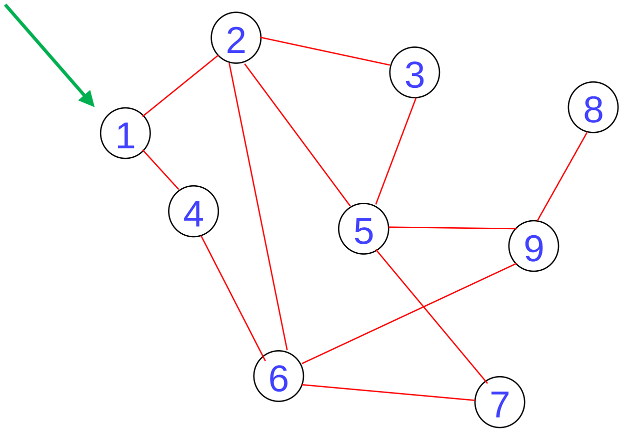
- Argument: In BFS, starting from vertex ***v***, the level number of vertex ***u*** is the length of the shortest path from ***v*** to ***u***
  - The shortest path is just the path which has the least number of edges from ***v*** to ***u***
  - First, to show that there is a path of length as the level number assigned by the BFS tree
    - Traverse though the predecessors upto the root

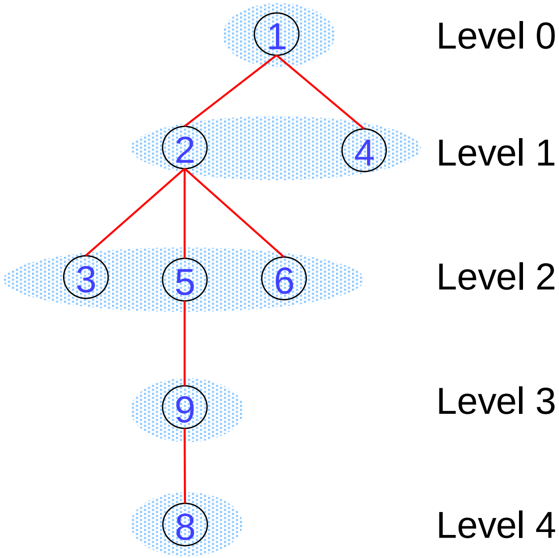




Application: Shortest Path using BFS

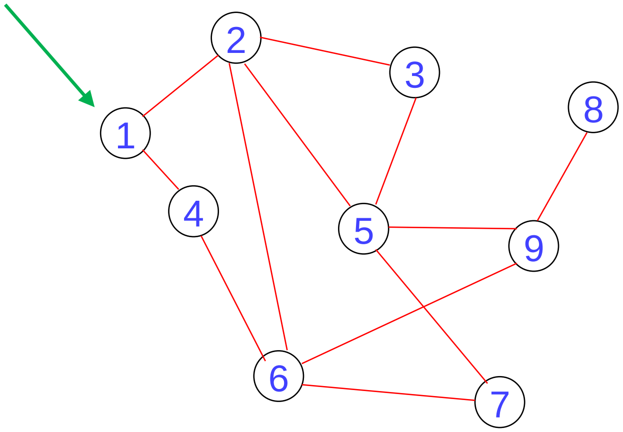
– Can there be a path of length one less than the level number assigned by the BFS tree or less
  - Then we will have to jump a level in the tree which violates the fact that this is a BFS
  - This is a partition it gives by the BFS
  - In BFS tree, level number = the length of the shortest path

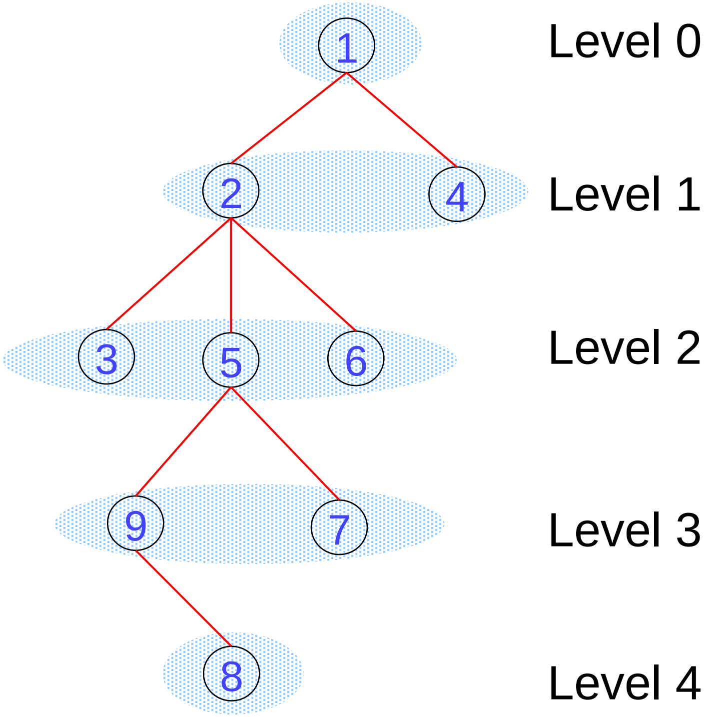




Application: Diameter of a Graph using BFS

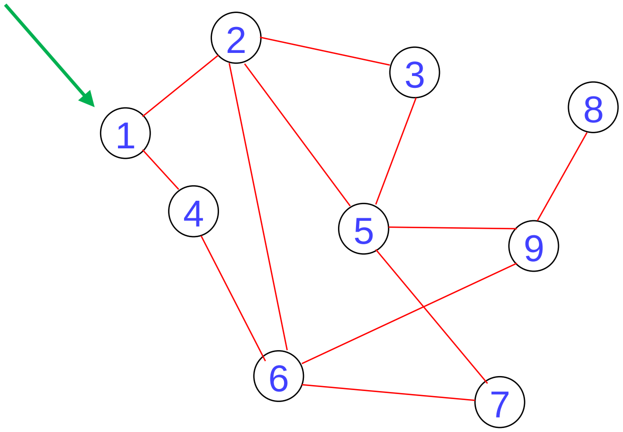
- Diameter(***G***) = Maximum *distance* between two vertices in ***G***
- *Distance* between two vertices equals length of shortest path (not between the longest or any path)
- What is the diameter of a graph ***G***?
  - Start a BFS from any vertex

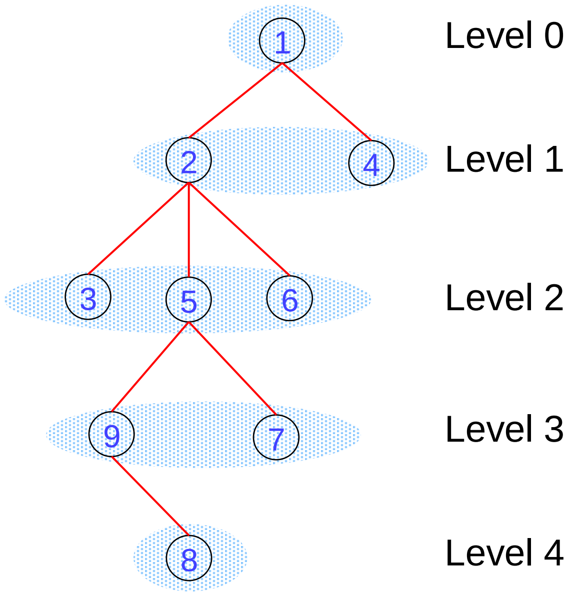




Application: Diameter of a Graph using BFS

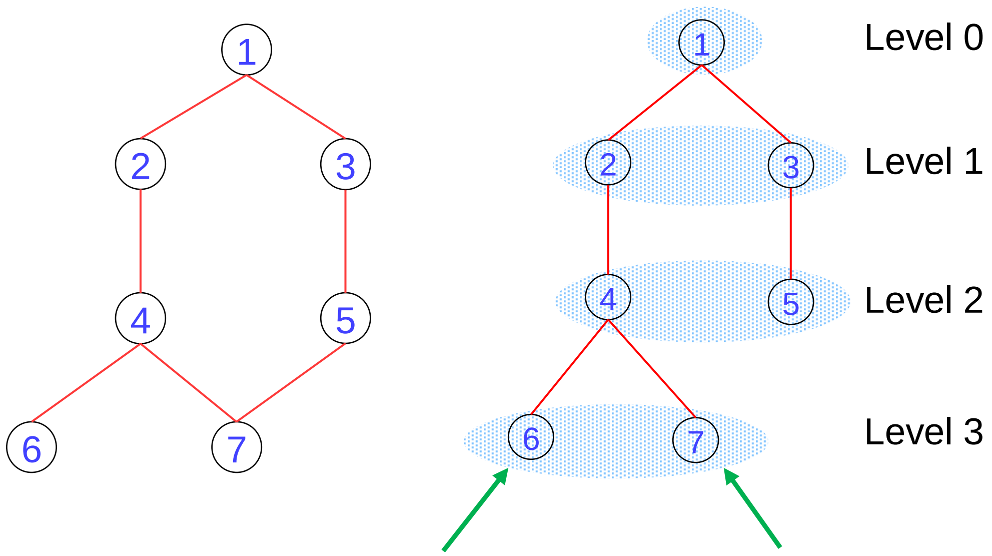
- Start a BFS from any vertex
- Diameter(***G***) ≤ 2 × the maximum level number for any BFS
- Diameter(***G***) ≥ the maximum level number for any BFS
- Then what? Do we need to do BFS from all other vertices taking each of them as root?





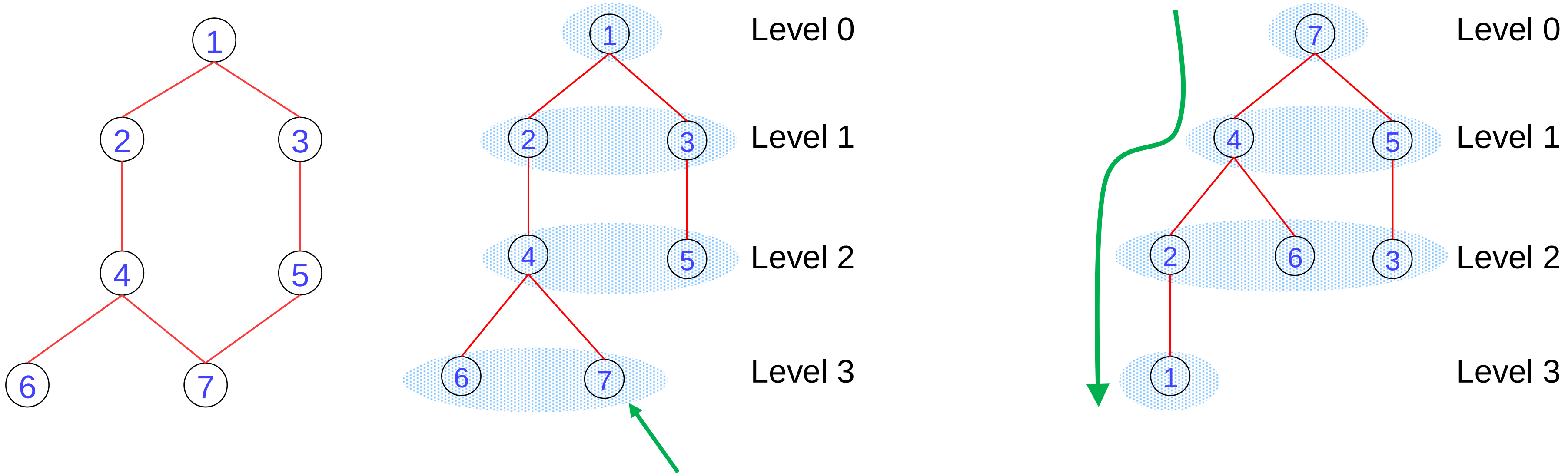
Application: Diameter of a Graph using BFS

- Why the 2-BFS solution will only work for the trees, not for a cyclic graph?



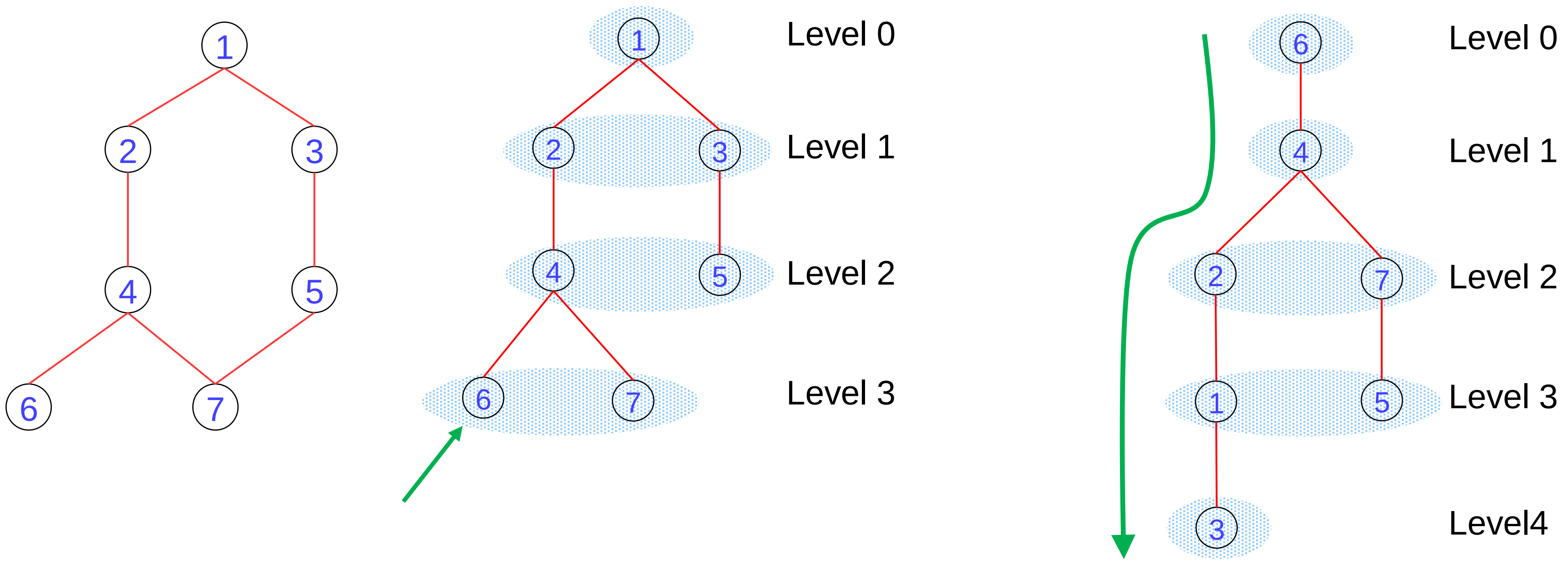
Application: Diameter of a Graph using BFS

- Why the 2-BFS solution will only work for the trees, not for a cyclic graph?



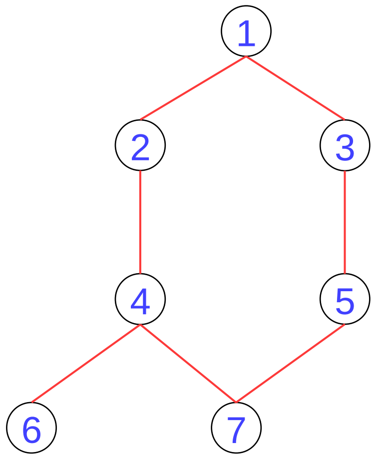
Application: Diameter of a Graph using BFS

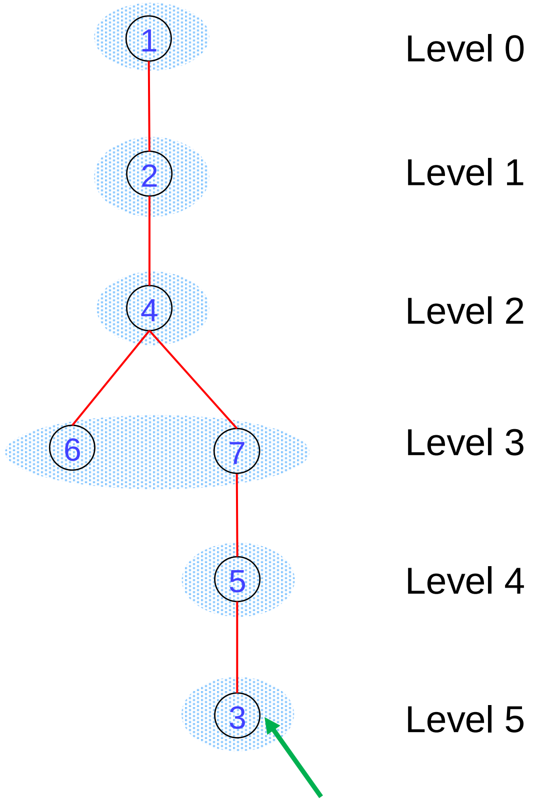
- Why the 2-BFS solution will only work for the trees, not for a cyclic graph?

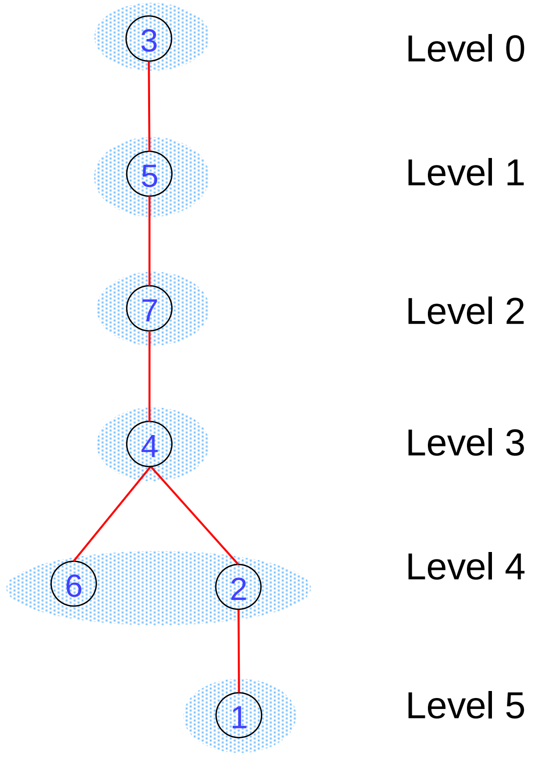


Application: Diameter of a Graph using DFS

- Why the 2-DFS solution will only work for the trees, not for a cyclic graph?







Next Lecture

Details of Depth-First Search

Thank you for your attention...

Any question?

Contact:
Department of Information Technology, NITK Surathkal, India
6th Floor, Room: 13
Phone: +91-9477678768
E-mail: shrutilipi@nitk.edu.in, shrutilipi.bhattacharjee@tum.de