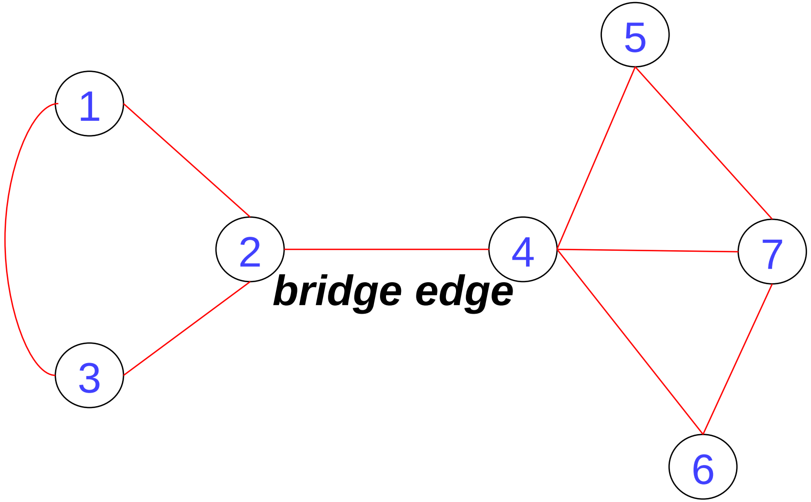
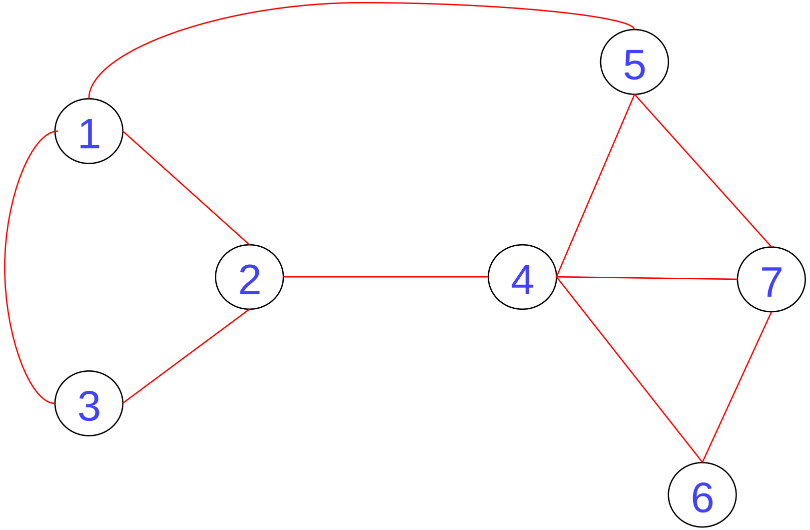
Applications of Depth-First Search

Application: 2-Edge Connected using DFS

- A graph is ***2-Edge connected*** iff it remains connected after the removal of any one edge

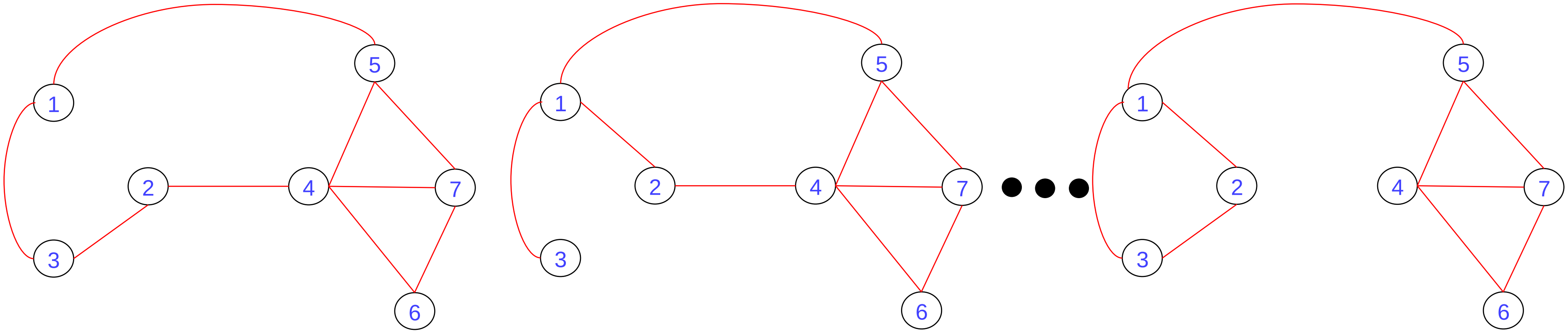
- Graph ***G*** = ***(V, E)*** is ***2-Edge connected*** iff ***G – {e}*** is still connected ***∀e ∈E***





Application: 2-Edge Connected using DFS

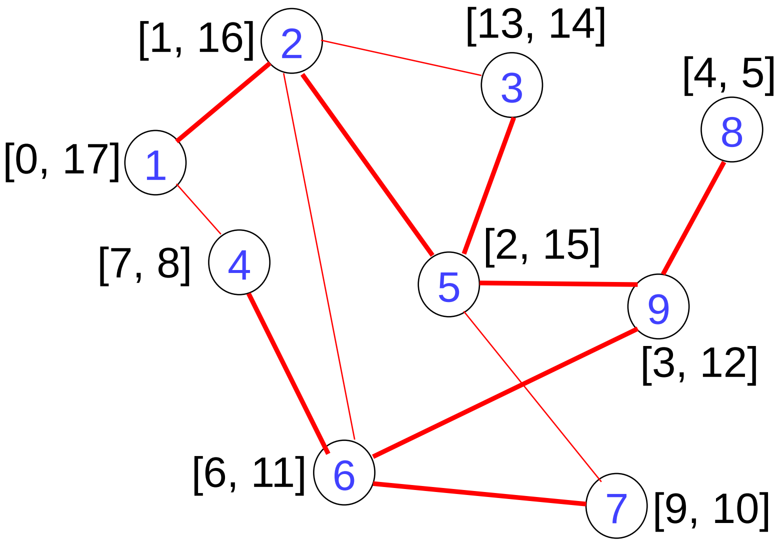
- Given a graph ***G*** = ***(V, E)***, how will you check if it is a ***2-Edge connected*** graph?
  - Can be done in $O(E^2)$ time by removing every edge and checking if the resulting graph is connected

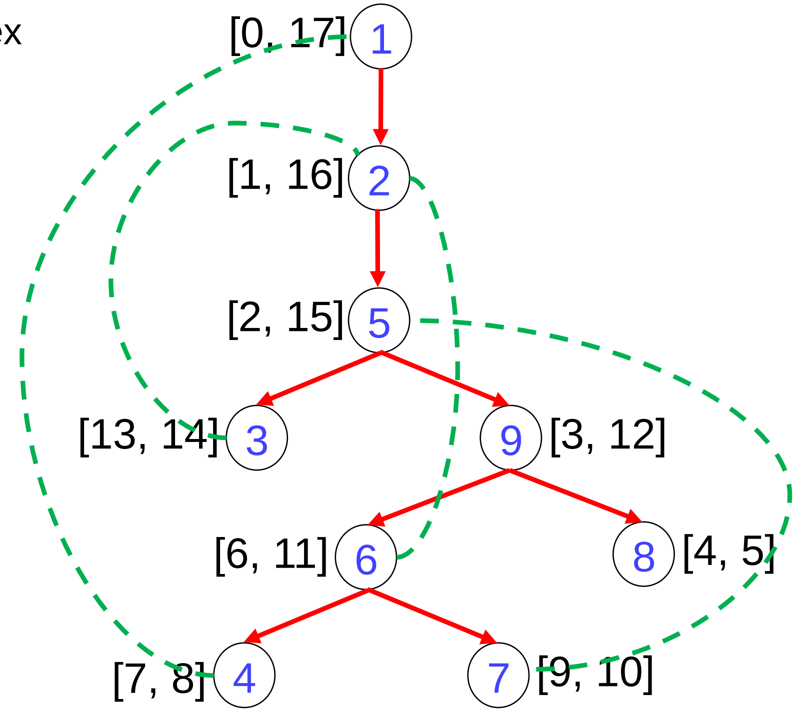


- How can we do it in linear time?

Application: 2-Edge Connected using DFS

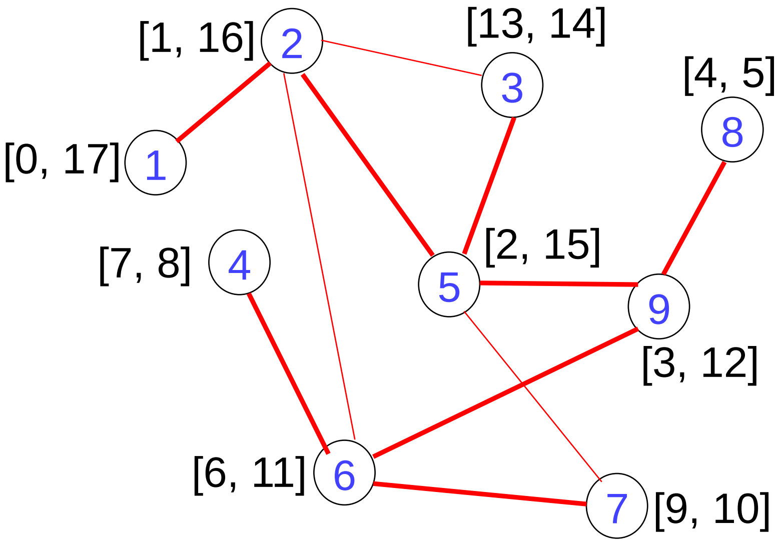
- Can be done using DFS
  - To check while backtracking from a vertex ***v***, the edge between this vertex and its immediate predecessor vertex (parent in DFS tree) is not a bride
  - How to ensure that it is not a bridge?
    - Atleast there is one ***back edge*** from ***v*** or the descendent tree of ***v*** to any of its ancestors

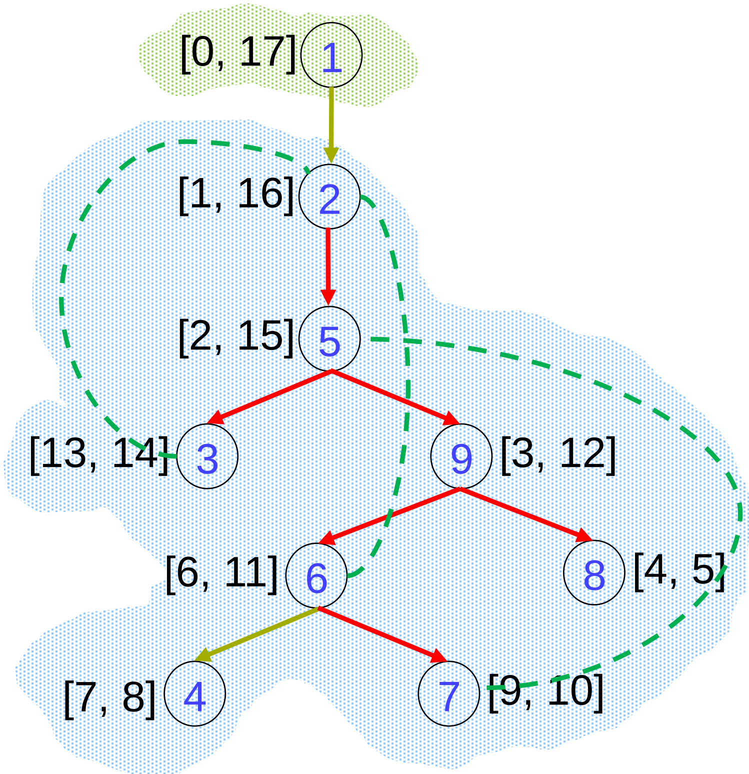




Application: 2-Edge Connected using DFS

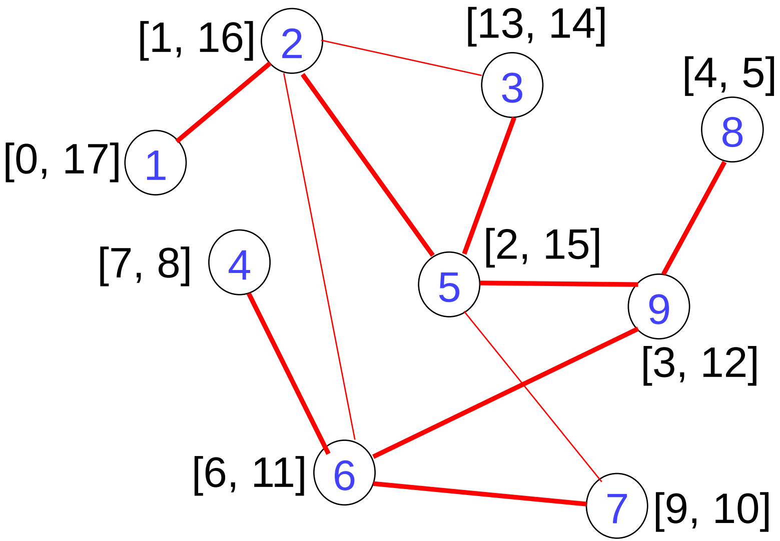
- Can be done using DFS
  - To check while backtracking from a vertex ***v***, the edge between this vertex and its immediate predecessor vertex (parent in DFS tree) is not a bride
  - How to ensure that it is not a bridge?
    - Atleast there is one ***back edge*** from ***v*** or the descendent tree of ***v*** to any of its ancestors

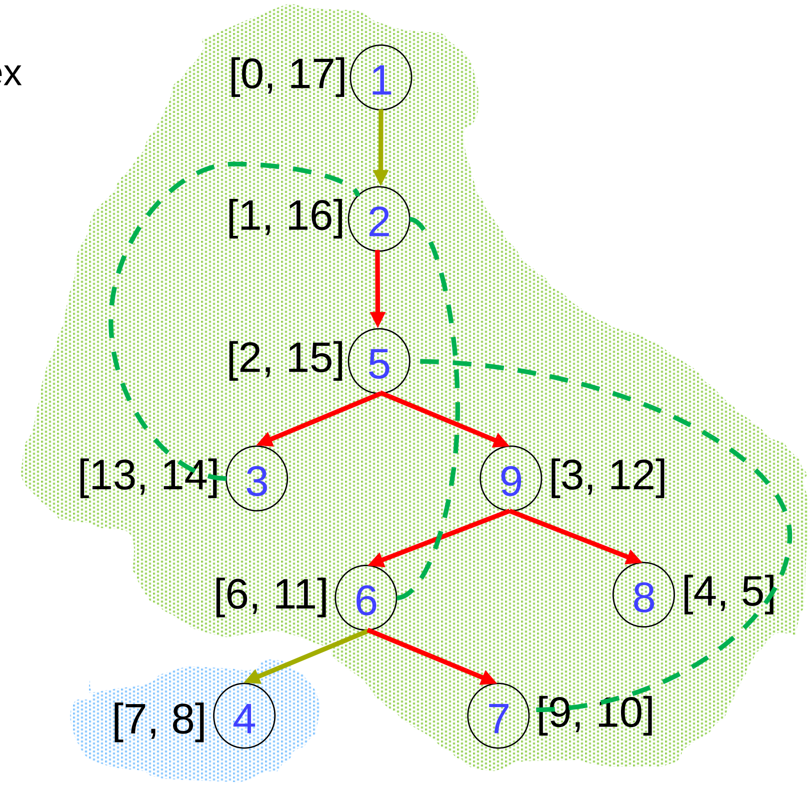




Application: 2-Edge Connected using DFS

- Can be done using DFS
  - To check while backtracking from a vertex ***v***, the edge between this vertex and its immediate predecessor vertex (parent in DFS tree) is not a bride
  - How to ensure that it is not a bridge?
    - Atleast there is one ***back edge*** from ***v*** or the descendent tree of ***v*** to any of its ancestors

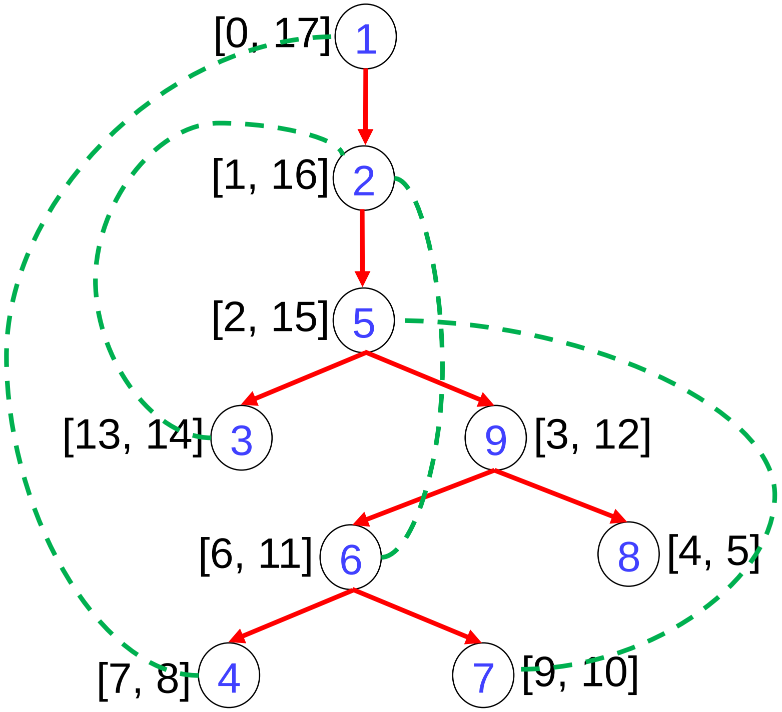




Application: 2-Edge Connected using DFS

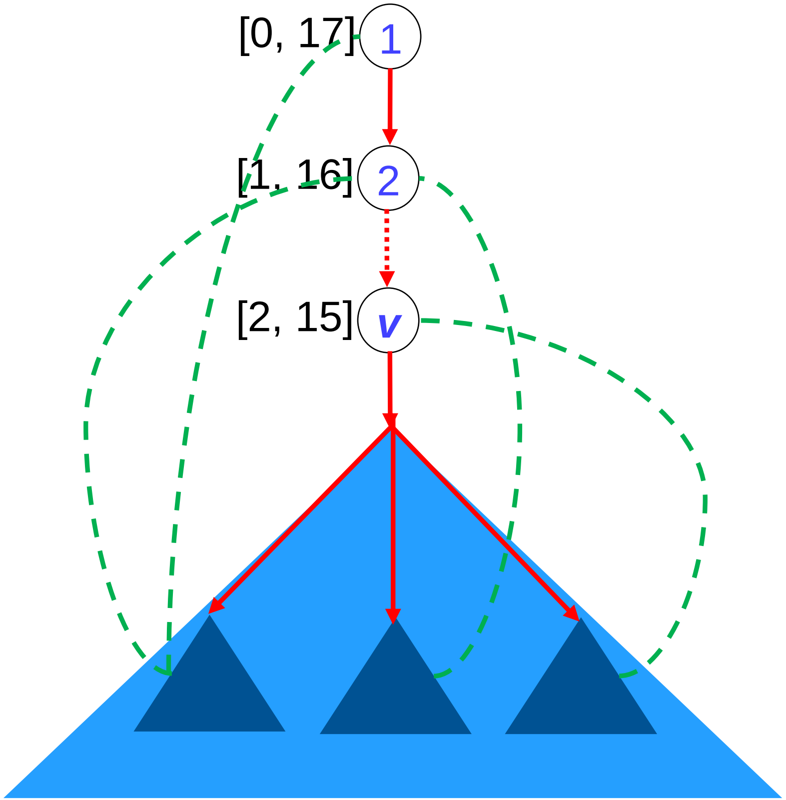
- Do we need to keep track of all back edges?
- If we keep track of every back edge then we are going to be spending a lot of time
- The back edge which is going to a node which is closest to the root is of our interest: ***deepest back edge***

- How can I figure out which is the ***deepest back edge***?
  - By looking at the arrival time of the other end point, as it is closest to the root



Application: 2-Edge Connected using DFS

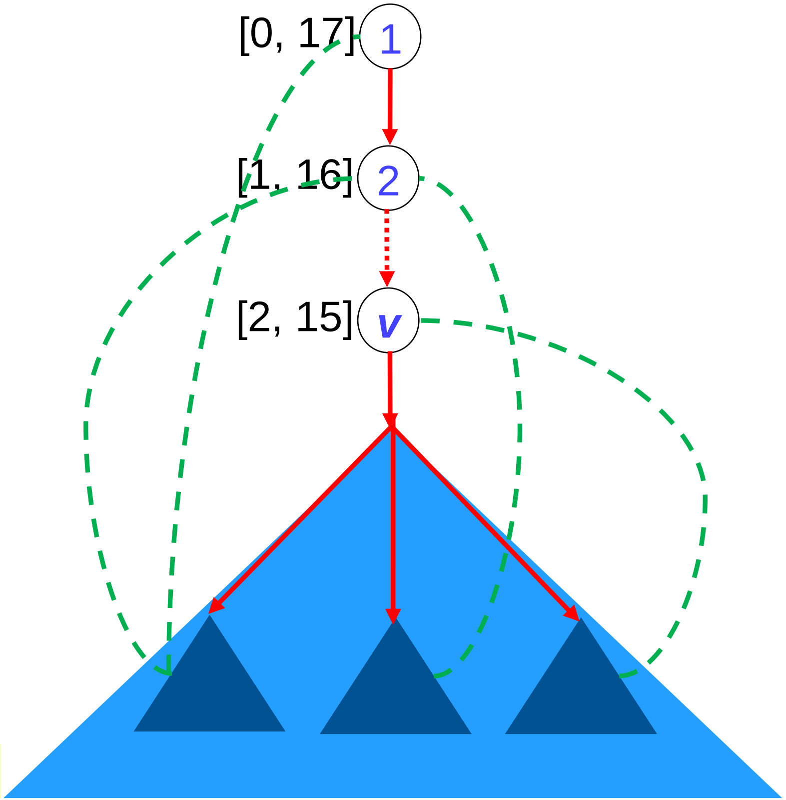
- ***Deepest back edge***
  - All the subtree rooted at ***v*** may have ***back edges*** which can be explored recursively using DFS
  - We must take the one with minimum arrival time (including ***back edges*** from ***v***)



Application: 2-Edge Connected using DFS

```
time = 0;
2EC(v)
{
   visited[v] = 1;
   arr[v] = time++;
   deepest_BE = arr[v];
   for (for all vertex u adjacent from v) do
      if !visited[u] then
          deepest_BE = min(deepest_BE, 2EC(u));
      else
          deepest_BE = min(deepest_BE, arr[u]);
if deepest_BE = arr[v] then
      return deepest_BE;
}
```

A bridge is found



Application: 2-Edge Connected using DFS

```
time = 0;
2EC(v)
{
  .....
  .....
     if !visited[u] then
        (v, u) is a tree edge;
        deepest_BE = min(deepest_BE, 2EC(u));
    else
       if !((v, u) is a tree edge)
       deepest_BE = min(deepest_BE, arr[u]);
  .....
  .....
}
```

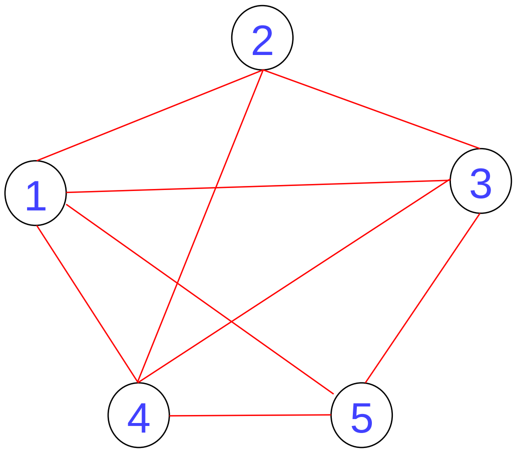
- Two special cases:
  - If (***v, u***) is a tree edge
  - If ***v*** is the starting vertex

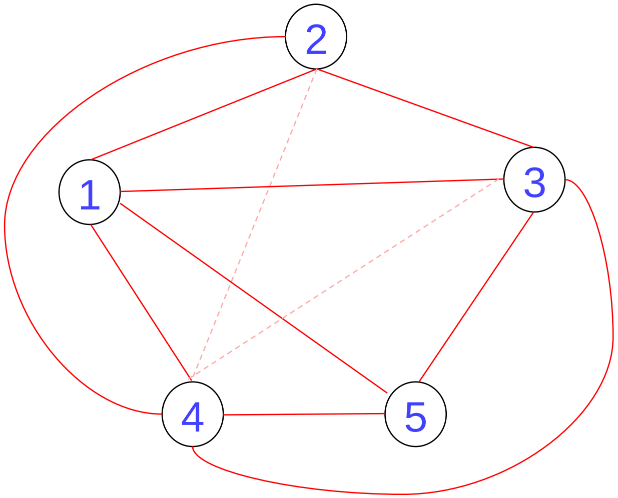
```
time = 0;
2EC(v)
{
  .....
  .....
  if deepest_BE = arr[v] && v != starting vertex then
     return deepest_BE;
}
```

- Time complexity
  - Same as DFS, ***O(V + E)***
  - If the graph is connected, then ***O(E)***

Application: Planner Graph using DFS

- Given a graph ***G* = *(V, E)***, how will you check if it is a ***planner graph?***
- A ***planar graph*** is a graph which can be drawn in the plane such that its edges do not intersect: planer drawing of a graph
- We can draw it whichever way I want but the edges should not intersect
- Is this graph a planer graph?

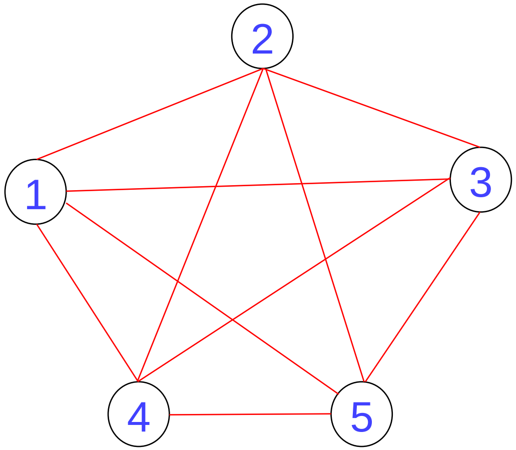


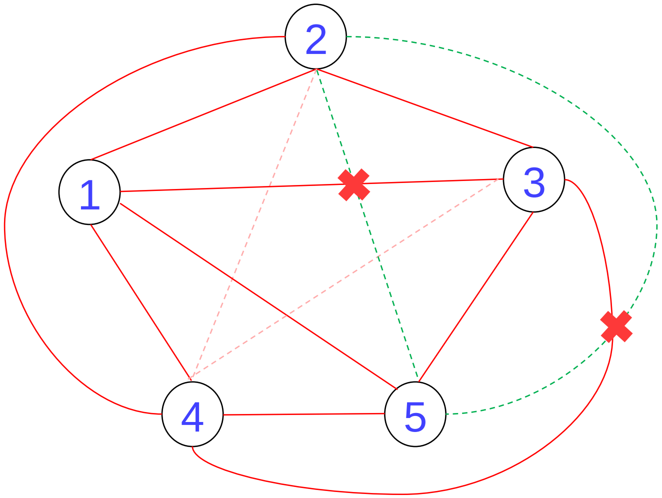


Application: Planner Graph using DFS

- Now let us consider the complete graph with 5 vertices
- Is this graph a ***planer graph***?

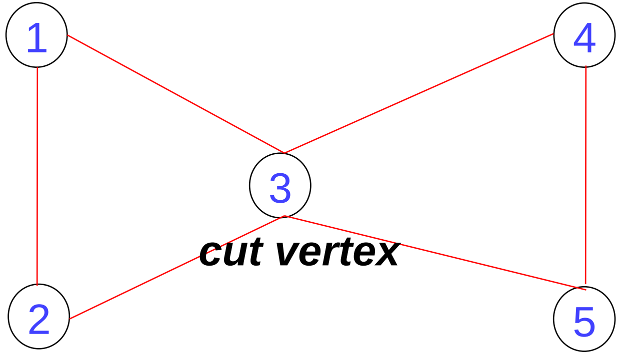
- DFS algorithm can be used to check if a given graph is a ***planer graph***





Application: 2-Vertex Connected using DFS

- A graph is ***2-vertex connected***, if removing any vertex still keeps the graph connected
- In removing a vertex, we also have to remove the edges incident to that vertex
- Is this graph ***2-edge connected***?
- Is this graph ***2-vertex connected***?
- DFS algorithm can be used to check if a given graph is a ***2-vertex connected***



Next Lecture

Depth-First Search in Directed Graphs

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