Details of Depth-First Search

DFS Algorithm: Revisited

```
depthFirstSearch(v)
{
  Label vertex v as discovered
  for (each undiscovered vertex u
        adjacent from v) do
    if vertex u is not labeled as
        discovered then
        depthFirstSearch(u);
}
```

```
depthFirstSearch(v)
{
  visited[v] = 1;
  for (for all vertex u adjacent from v) do
    if !visited[u] then
        depthFirstSearch(u);
}
```

Colors in DFS

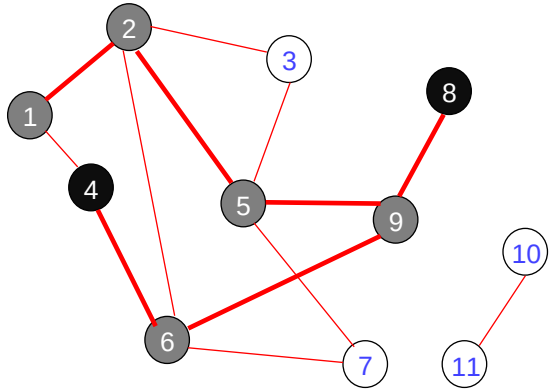
- ***White:*** The vertex ***v*** is not yet discovered
- ***Gray:*** The vertex ***v*** has already been discovered, but all its adjacent vertices are not yet discovered, hence the vertex ***v*** is still in the stack
- ***Black:*** The vertex ***v*** is already pop out of stack, hence discovered and finished
- ***visited[]*** is an array of size ***N***, consists of elements either ***0*** (white) or ***1*** (black and gray)

Visit/mark/label start vertex and put in a LIFO stack ***S***

1	2	5	9	8	6	4				

Visited[ ] =

1	1	0	1	1	1	0	1	1	0	0



Pre- and Post-visited Times in DFS

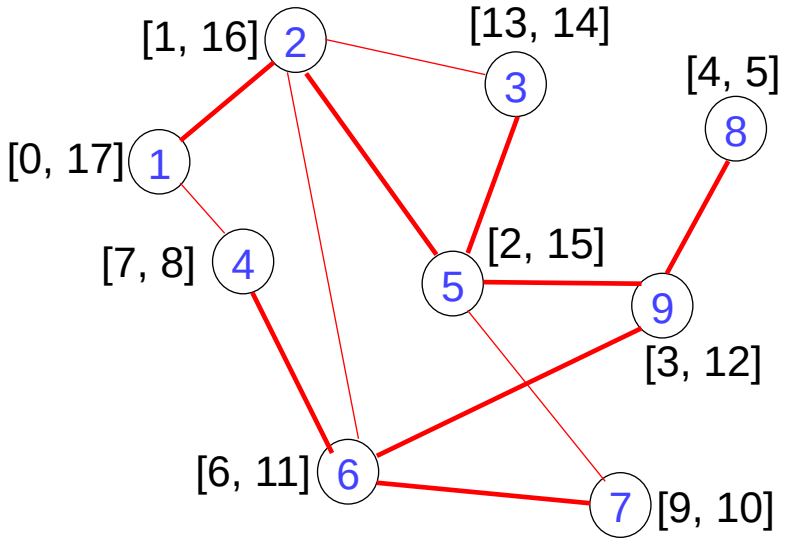
- *Pre-visit and Post-visit* numbers are the extra information that can be stored while running a DFS on a graph and which turns out to be really useful
- *Pre-visit* number tells the time at which the vertex is marked visited
- *Post-visit* number tells the time at which the vertex processing is finished
- It can be thought of as a counter

Pre-visit/ arrival time[ ] =

0	1	13	7	2	6	9	4	3

Post-visit/ departure time[ ] =

17	16	14	8	15	11	10	5	12



DFS Algorithm: Modified

Visited[ ] =

1	1	0	1	1	1	0	1	1

Pre-visit/ arrival time (arr[ ]) =

0	1	13	7	2	6	9	4	3

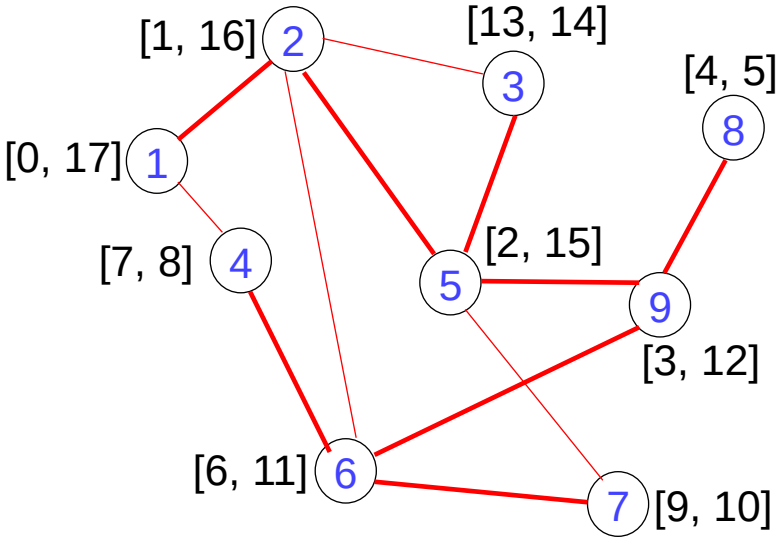
Post-visit/ departure time (dept[ ]) =

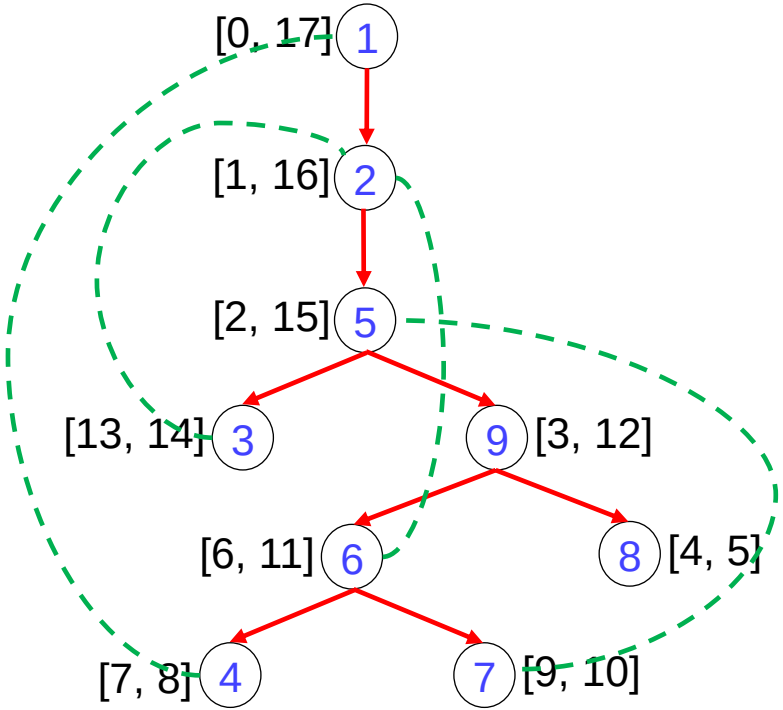
17	16	14	8	15	11	10	5	12

```
time = 0;
depthFirstSearch(v)
{
  visited[v] = 1;
  arr[v] = time++;
  for (for all vertex u adjacent from v) do
    if !visited[u] then
        depthFirstSearch(u);
  dept[v] = time++;
}
```

DFS Tree for Undirected Graph

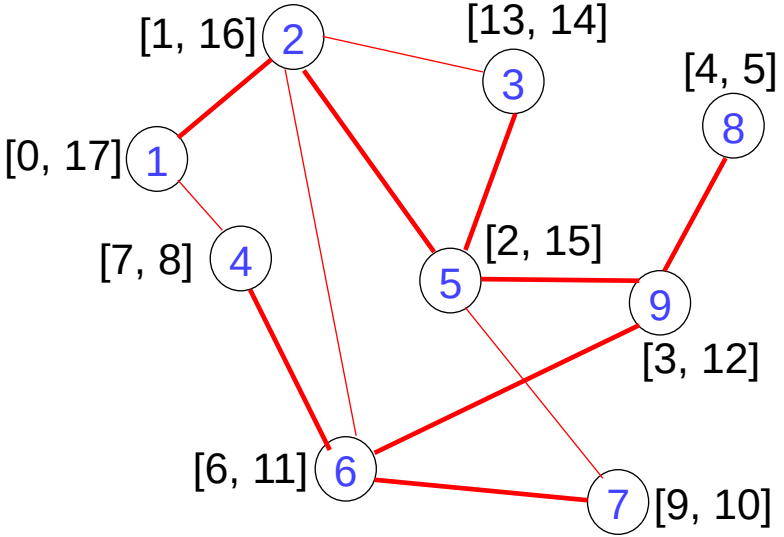
- DFS tree gives us the notion of ancestors and descendants (natural parent-child relation)

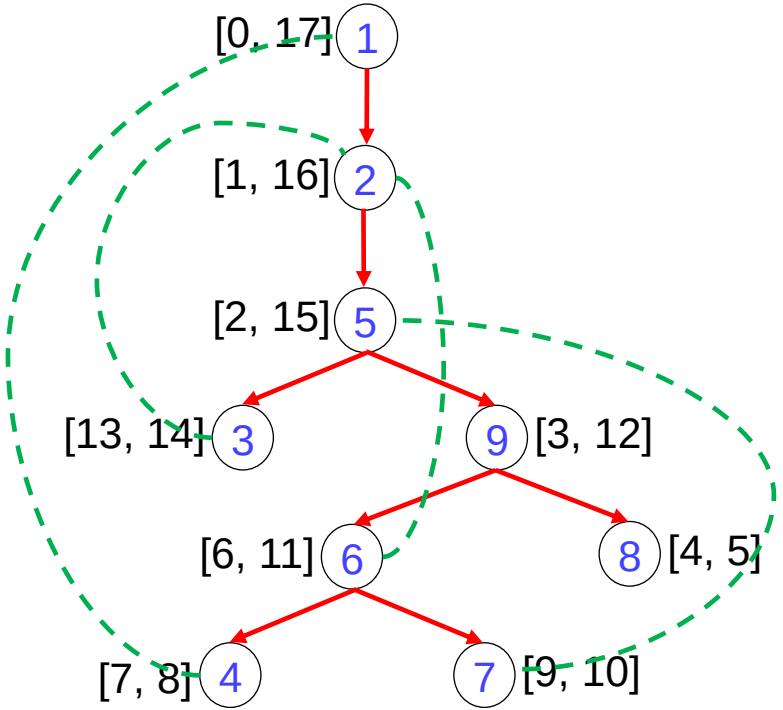




DFS Tree for Undirected Graph

- Red edges defines the DFS tree, and are called ***tree edges***
- The green edges are called ***back edges***
- Why not ***front edges***? Because it is going back to a vertex which is already visited

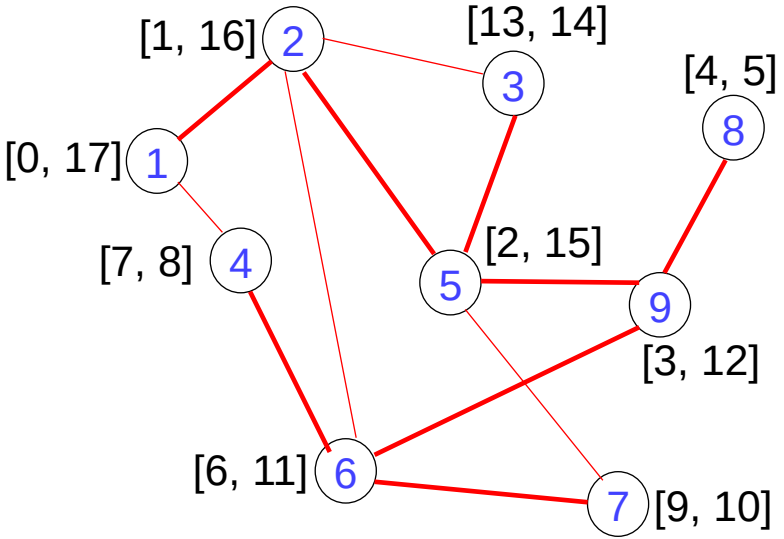


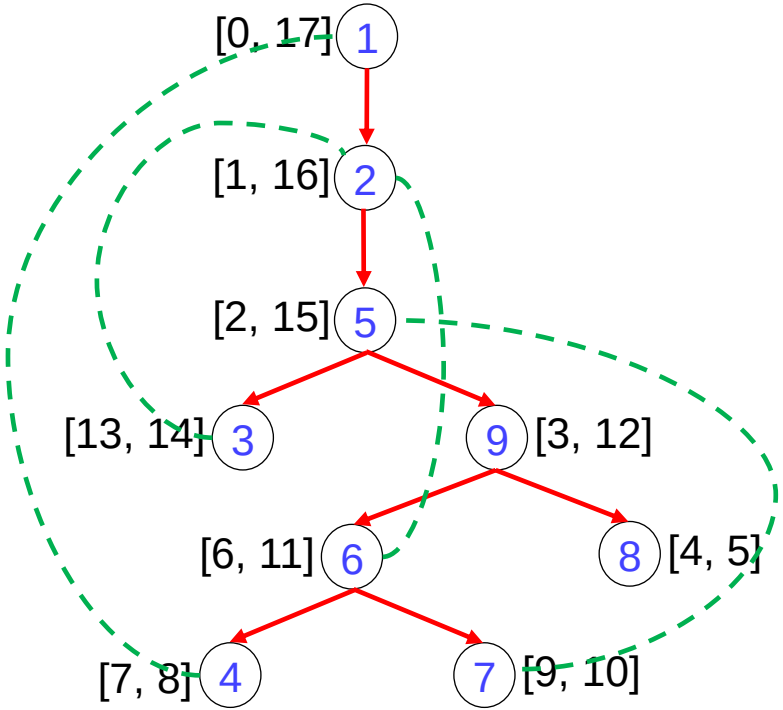


- Note: Remember this concept is for undirected graph

Back Edge

- An edge from a vertex to an ancestor in the DFS tree is called a ***back edge***
- Why ***back edge*** is not possible from a vertex to another vertex which is not an ancestor?
- DFS classifies every edge to be a **tree edge** or a **back edge**





- Note: Remember this concept is for undirected graph

DFS Algorithm: Modified

Visited[ ] =

1	1	0	1	1	1	0	1	1

Pre-visit/ arrival time (arr[ ]) =

0	1	13	7	2	6	9	4	3

Post-visit/ departure time (dept[ ]) =

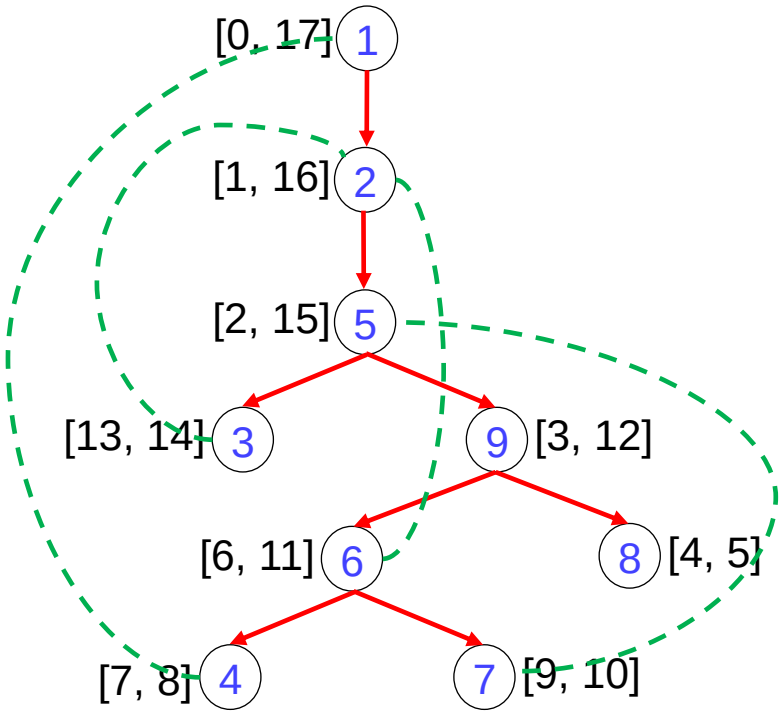
17	16	14	8	15	11	10	5	12

```
time = 0;
depthFirstSearch(v)
{
  visited[v] = 1;
  arr[v] = time++;
  for (for all vertex u adjacent from v) do
    if !visited[u] then
        depthFirstSearch(u);
        (v, u) is a tree edge;
  dept[v] = time++;
}
```

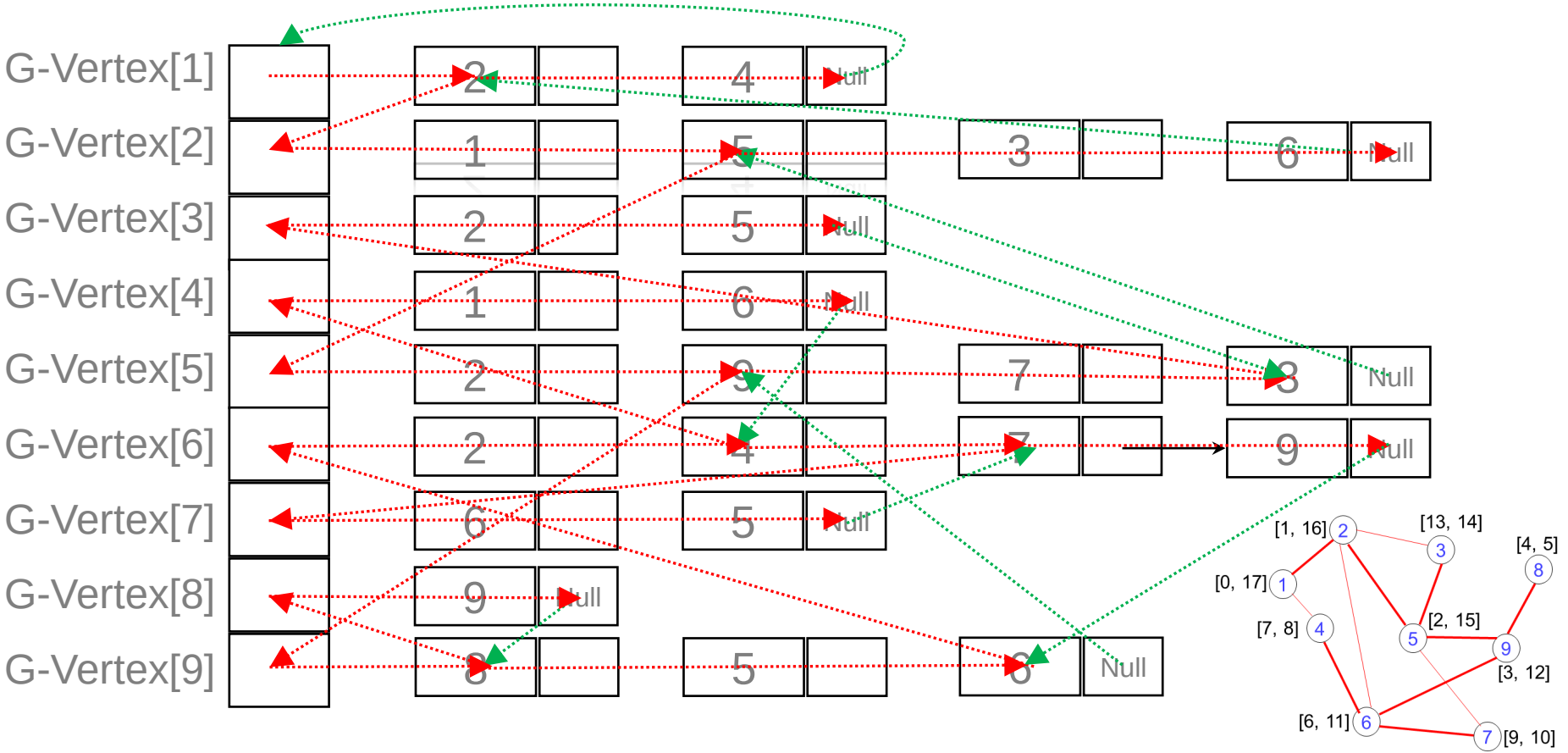
Edge Type and Pre/Post-visited Times

- For any (***u, v***) ∈ ***E***, if (***u, v***) is a ***back edge***
  - ***u*** is an ancestor of ***v***
  - ***v*** will be finished before ***u***
  - Then arr[***u***] < arr[***v***] and dept[***u***] > dept[***v***]

- For any (***u, v***) ∈ ***E***, if (***u, v***) is a ***tree edge***
  - ***u*** is an ancestor of ***v***
  - ***v*** will be finished before ***u***
  - Then arr[***u***] < arr[***v***] and dept[***v***] < dept[***u***]

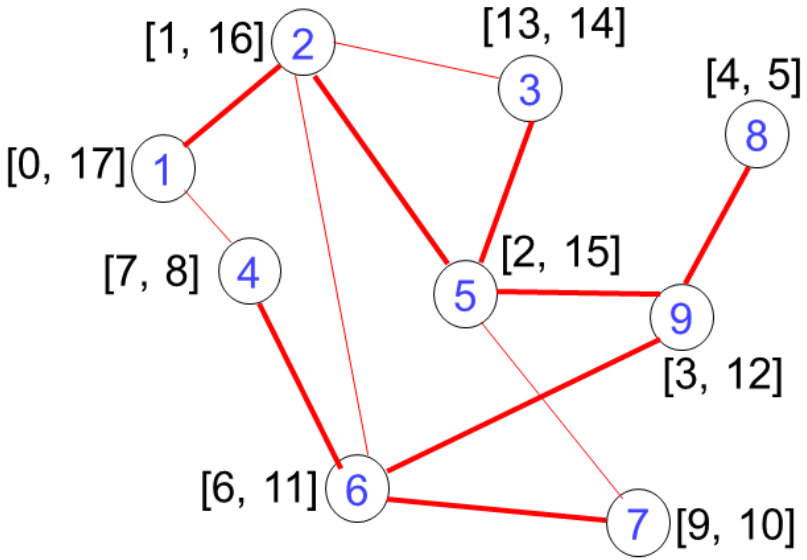


DFS Algorithm Time Complexity



DFS Algorithm Time Complexity

- Every edge is visited twice, and exactly twice
- Time complexity: ***O*(*V* + *E*)**
- If the graph is connected, then ***E* ≥ *V* – *1***
- Time complexity becomes ***O*(*E*)**



Next Lecture

Applications 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