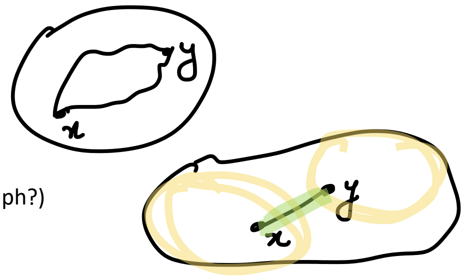
Screen-Recorder / Proctoring ← video on

compulsory

Depth First Search (DFS)

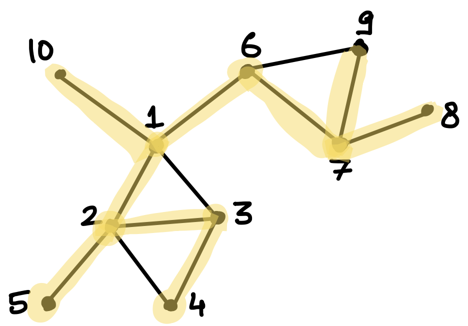
Applications:

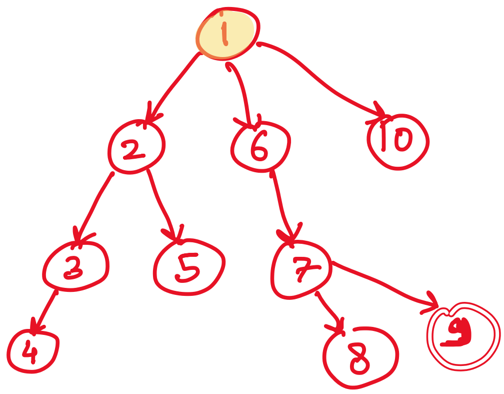


- **Biconnected** components of a graph. ✓
  (Are there two edge-disjoint-paths between each vertex-pair ?)
- Finding **bridges** in a graph. ✓
  (List all those edges e for which failure of e in G disconnects the graph?)
- **Planarity testing** of a graph
  (Can a given graph be embedded in a plane ?)
- **Strongly connected** components of a directed graph. ✓
  (the extension of connectivity in case of directed graphs)

In BFS: we explore layer by layer, so each vertex explores all neighbors.

In DFS: explores as far as possible along each branch and then back trace.





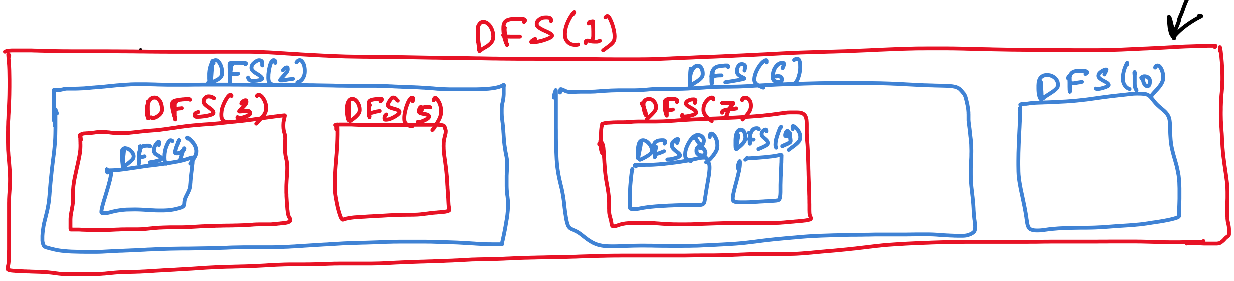
Recursive Code: Initially mark all vertices in G as "*unvisited*".

set – 1 starting node

DFS(x):

- Mark x as "*visited*".
- For each neighbour y of x:

  If y is "unvisited" then 1) set y as child of x, and 2) invoke **DFS**(y).



Lemma 1: *The vertices visited during recursive call DFS(x) are descendants of x in DFS tree*

.

Proof: To prove the claim we will apply induction on the depth of DFS tree and proceed in a bottom-up manner.

Hyp(i) : For each vertex x at depth "i" in the DFS tree, we have:

"vertices visited by DFS(x) = descendants of x in the DFS tree"

Hyp(i) => Hyp(i-1) :
Take a vertex "x" at depth "i-1". We have two cases.

Case 1: x is a leaf node: In this case DFS(x) only visits itself as all its neighbors are already visited. So vertices visited by DFS(x) = {x} = descendants of x in the DFS tree.

Case 2: x is a internal node: Let "x1, …, xk" be children of x. These children have depth "i". Observe that in the recursive call of DFS(x), we visit x and invoke DFS(x1),…,DFS(xk).

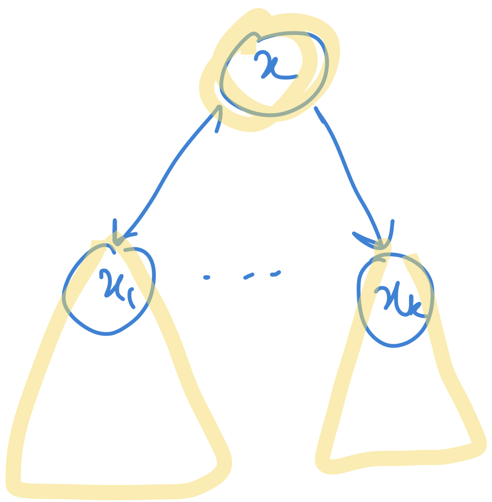
Therefore, vertices visited by DFS(x) = {x} + vertices visited by DFS(x1), … ,DFS(xk).

By applying induction hypothesis on vertices x1,…,xk lying at depth "i", we get:

vertices visited by DFS(x) = {x} + descendants of x1, … ,xk in the DFS tree.

Since right-side term in above expression is just descendants of x (why?), we get following:

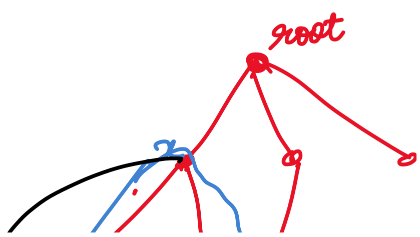
vertices visited by DFS(x) = descendants of x in the DFS tree.



Lemma 1: *The vertices visited during recursive call DFS(x) are descendants of x in DFS tree* .

G \ T

Lemma 2: *Let T be a DFS tree of G=(V,E), and (x, y) be a non-tree edge of G satisfying x,y are vertices in T. Then one of x or y is an ancestor of the other.*

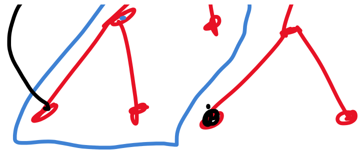


Proof: Suppose x is visited before y in DFS traversal.

Before function DFS(x) returns, it must have visited y.

(Why? - bcoz y is neighbour of x).

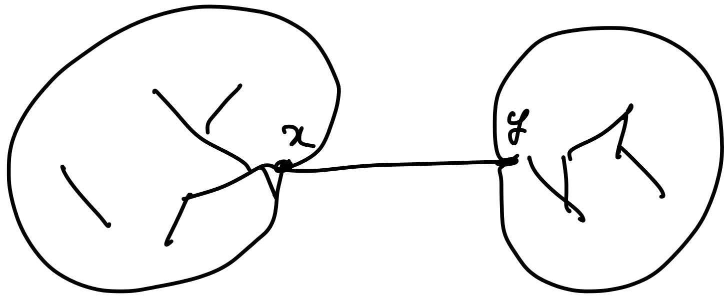
So by Lemma 1, y must be a descendant of x in the DFS tree.



Each non-tree edge (x,y) satisfy the cond^n

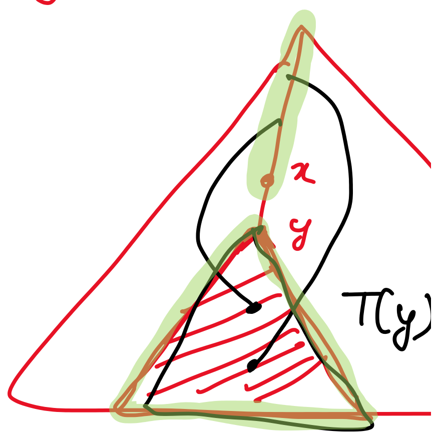
" $\exists$ ancestor - descendant relation b/w x & y"

Bridge Edge: *An edge (x,y) is said to be a bridge edge if x and y are disconnected in G\ (x,y).*



Q1. If G is connected and (x,y) is bridge edge, then implies (x,y) is a tree-edge.

Q2. If x is parent of y in DFS tree, and (x,y) is br
then $\underbrace{T(y)}_{\text{descendants of } y}$ has no non-tree edges that conn



How Q1, Q2 imply

$O(\underbrace{m}_{\text{no of edges in G}} + \underbrace{n}_{\text{vertices}})$ time algo to find ALL Bridge edges.