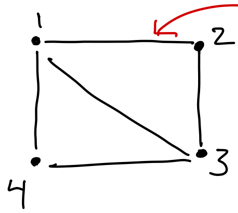
Graphs

- A **Graph** $G = (V, E)$ consists of a set of vertices and Edges. The edges represent a relation between 2 vertices:

$$E \subseteq V \times V$$

EX



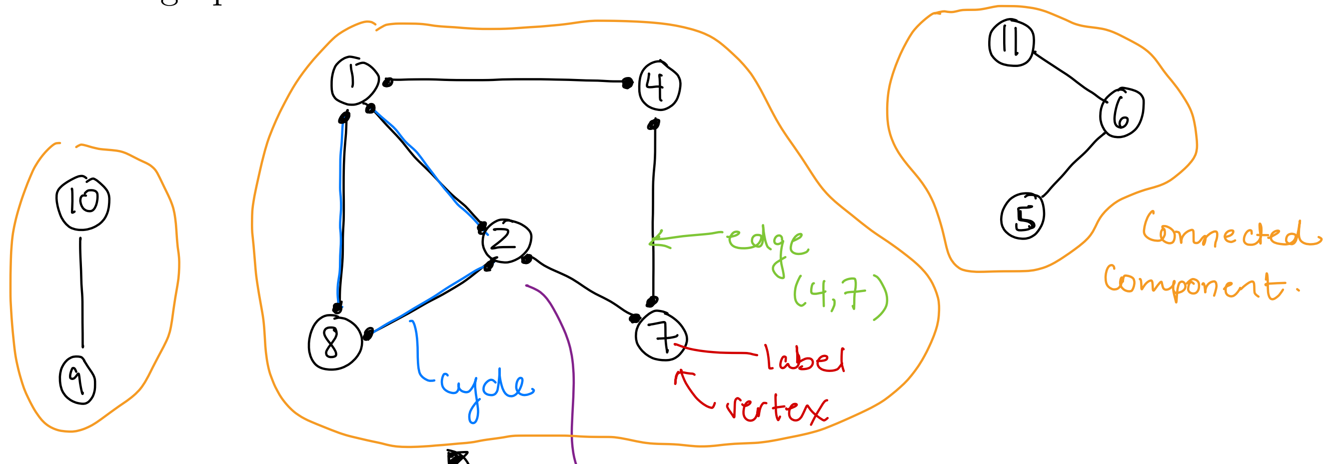
edge (1,2)

$V = \{1,2,3,4\}$.

- Definitions related to graphs:

EX

A graph with 3 connected components.



degree of ② is # of neighbours.
$= 3$.

Sum of degrees in this component
$= 2+3+2+2+3 = 12 = 2|E|$.

- What happens if we sum up all the degrees? Notice the number of "dots" above:

$$\boxed{\sum_{u \in V} \deg(u) = 2|E|}$$

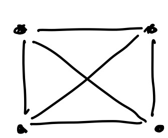
(Each edge is counted twice).

- Complete Graph:

$$|E| = \binom{n}{2}$$

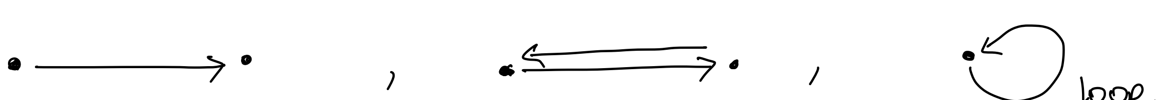
Every edge between 2 vertices is included.

ex



complete graph on 4 vertices.

- Directed Graph:

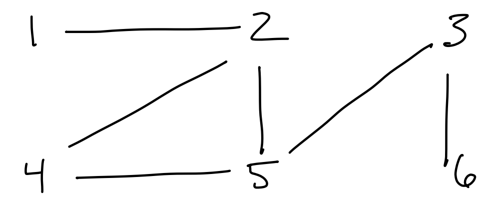


edges have a direction

Storage Options Graphs:

1. Adjacency matrix

Graph:



matrix of $|V|$ by $|V|$: 0: no edge, 1: edge.

	1	2	3	4	5	6
1	0	1	0	0	0	0
2	1	0	0	1	1	0
3	0	0	0	0	1	1
4	0	1	0	0	1	0
5	0	1	1	1	0	0
6	0	0	1	0	0	0

only need this information to recreate the graph!

diagonal entries are 0.

Notice that the matrix is symmetric and has 0's along the diagonal. So we only need to store:

Upper triangle written as sequence:
$$\underbrace{1\ 0\ 0\ 0\ 0}_{1}\ \underbrace{0\ 1\ 1\ 0}_{2}\ \underbrace{0\ 1\ 1}_{3}\ \underbrace{1\ 0}_{4}\ \underbrace{0}_{5}.$$

This number uniquely represents the undirected graph! $= (16796)_{10}$.

Questions...

1. Is there an edge from 2-5?

IF $|V| = n$ then the position of the bit for edge (u,v) is:

$$\underbrace{1+2+\cdots+(n-2)}_{\substack{\text{block for vertex } 2 \\ =10}} - \underbrace{(v-u-1)}_{\substack{5-2-1 \\ =2}} = \underbrace{\frac{(n-2)(n-1)}{2} - (v-u-1)}_{=8}.$$

ex $(16796)_{10} \bmod 2^8 = (10011100)_2$

bit for edge $2 \to 5$.

2. How to remove the edge 2- 5?

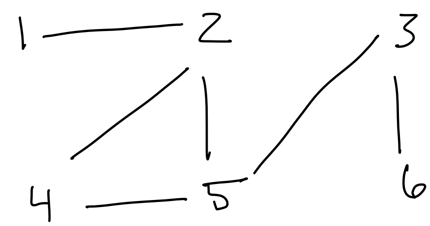
$$(16796)_{10} - 2^7 = 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0$$

bit is changed to 0.

3. Exercise: how to find all the neighbours of a node, using the binary representation.

2. Adjacency List

1 → 2
2 → 1 → 4 → 5
3 → 5 → 6
4 → 2 → 5
5 → 2 → 3 → 4
6 → 3



Each vertex has a linked list to the neighbours.

How does the storage compare?

- matrix: $O(|V|^2)$
- List: $O(|V|) + O(|E|)$

Each edge is stored twice in undirected graph.

note: $|E| \le \binom{|V|}{2}$.

Depth First Search (DFS)

Text book C22.

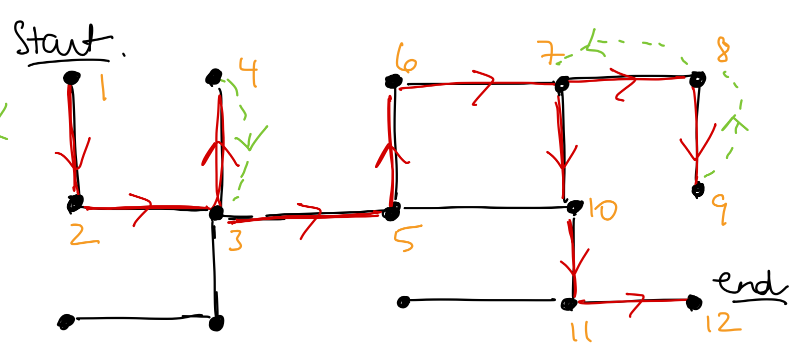
Algorithm for traversing/searching a graph. Originally introduced for solving mazes.

Intro animation: (on a maze with start/end).

Red: forward

Green: Backtrack

yellow: order of search.



idea: goes "as far" as it can until it must backtrack and find unexplored path

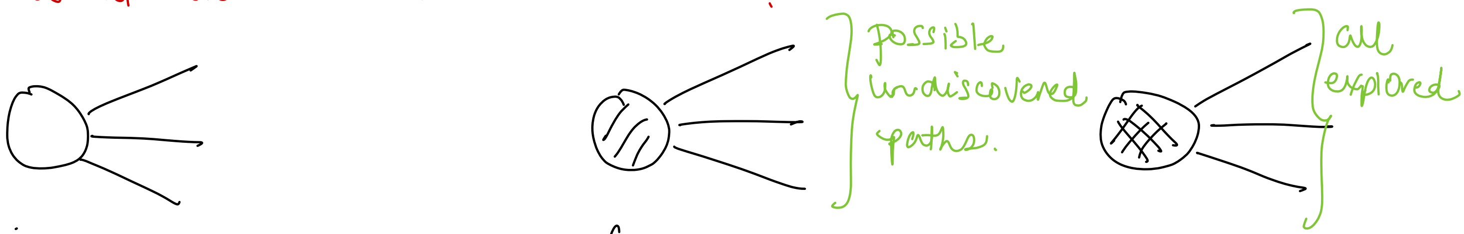
Before looking at the exact algorithm, let look at the information we will store along the way. We need to keep track of :

- parent pointers?
- time steps?
- if a node is visited?

For each vertex v, define:

- p[v]: pointer to the parent of v in the resulting DFS tree.
- d[v]: the "time" at which v was discovered.
- f[v]: the 'time' at which the seach is finished exploring paths out of v.
- colour[v]: white / grey / black.

What do the colours mean?



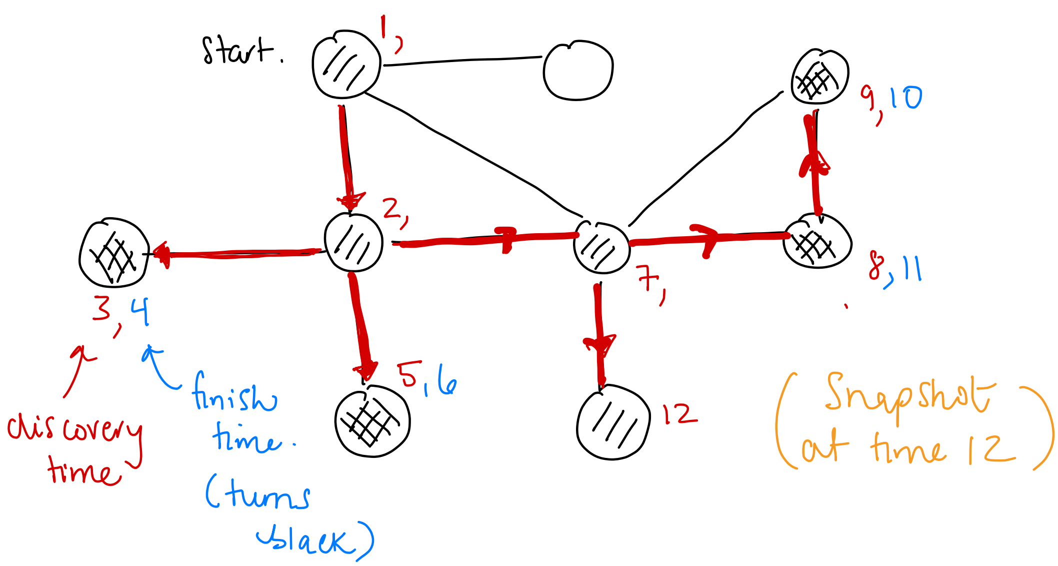
White: DFS has not yet discovered this node.

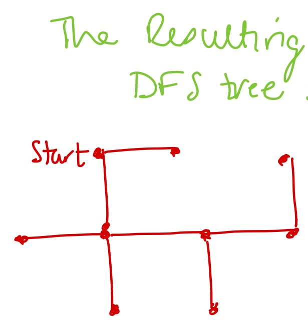
Grey: node is discovered, but not finished

Black: node finished.

The vertices all start out as white. The time is 0. The above properties are updated in DFS in the following way:

- Illustrate the algorithm on the graph below.
- Continues until all vertices are finished (black).





The **Algorithm:**:

initialize in $O(|V|)$ time.

time = 0 ← time variable

For all vertices $v \in V$

  colour[v] = white

  p[v] = nil.

given a start node.

DFS (u):

$O(|V|)$ overall.

  time = time + 1

  d[u] = time ← node u has been discovered.

  colour[u] = grey.

Executed $O(|E|)$ times

  $\forall$ v adjacent to u:

    if colour[v] = white

$p[v] = u$.

DFS(v).

O(|V|) overall.

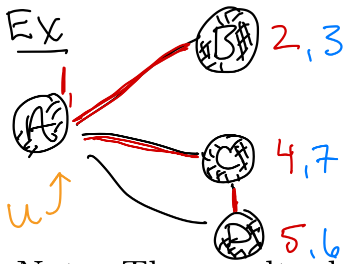
time = time + 1

colour[u] = black

f[u] = time.

node u is now "finished".

Ex



u's neighbours are B, C, D. But by the time the loop reaches vertex D, it is no longer white.

Note: The results depend on the order in which the neighbours are visited.

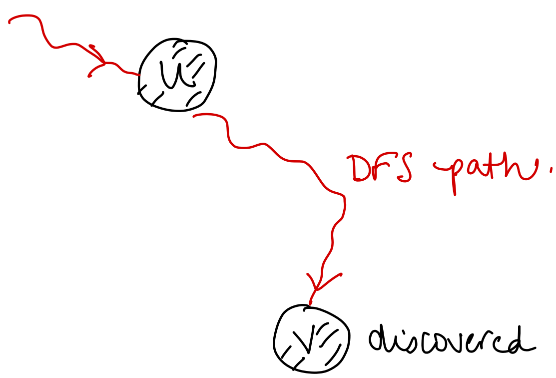
Complexity:

- Initialize O(|V|)
- Constant work per node (green above): O(|V|)
- Work done over the edges: (yellow above): O(|E|).

Total: O(|V| + |E|).

Properties of the DFS algorithm:

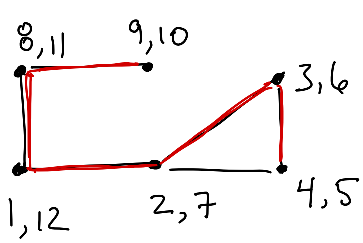
1. Vertex v is a descendent of vertex u if and only if it was discovered while u was grey:



Since u is grey, DFS is still exploring paths "out of" u. And thus v must be a descendent of u.

discovered for the first time by DFS!!

2. Nested Property: the intervals $(d[u], f[u])$ are either:

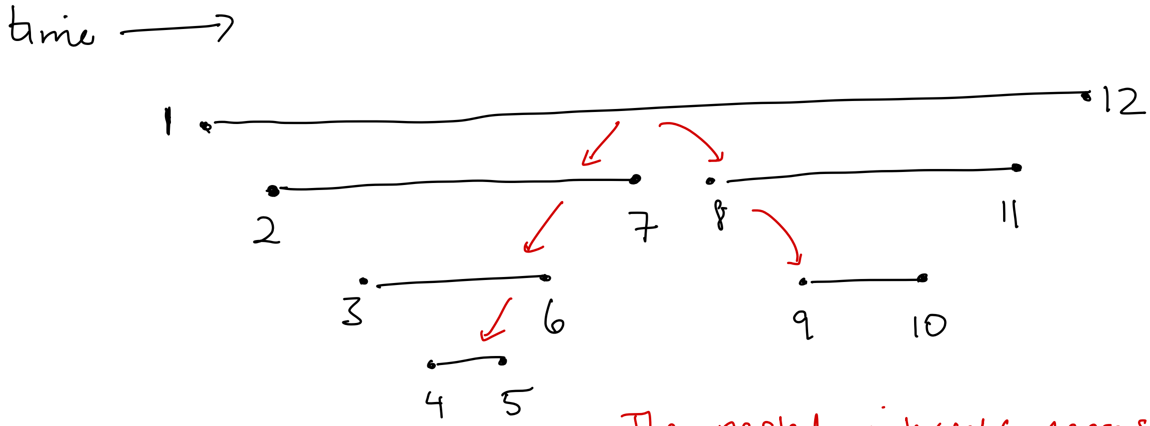


Nested! OR Disjoint.

Represents descendent.

We can draw these intervals on the time line:

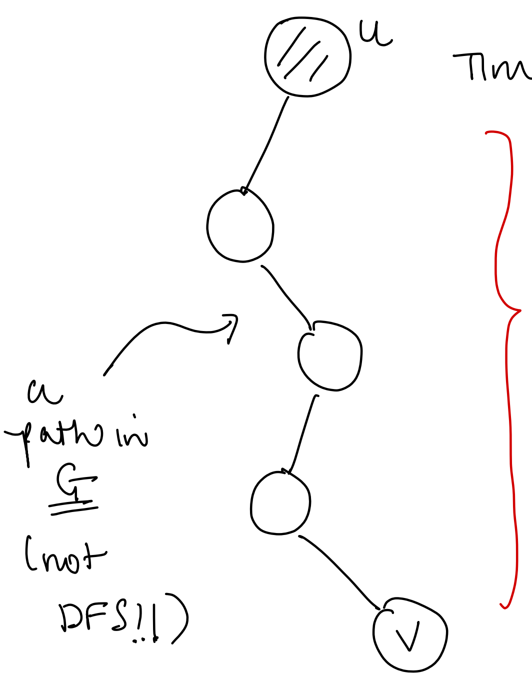
For the example above:



The nested intervals represent children in DFS tree.

3. White path theorem:
Vertex v is a descendent (DFS tree) of u if and only if at time $d[u]$ there is a white path from u to v in the graph.

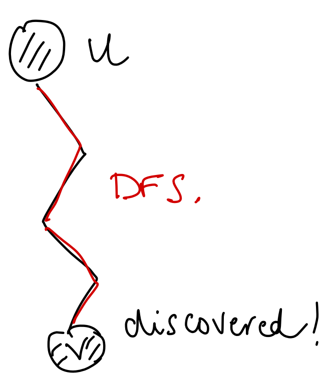
Time is d[u].



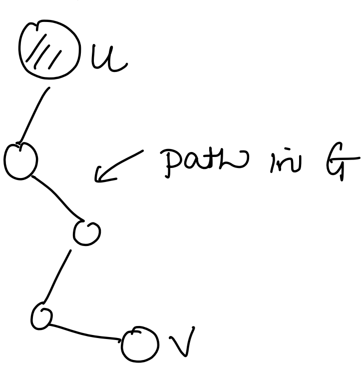
DFS will proceed from u, and will either follow this white path to v, or another white path to v, when it finds v, it will be a descendent of u.

* Both properties 1, and 3 are used to determine if v is a descendent of u.

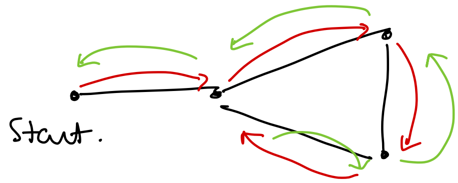
PROP 1) Time is d[v]



PROP 3) Time is d[u]

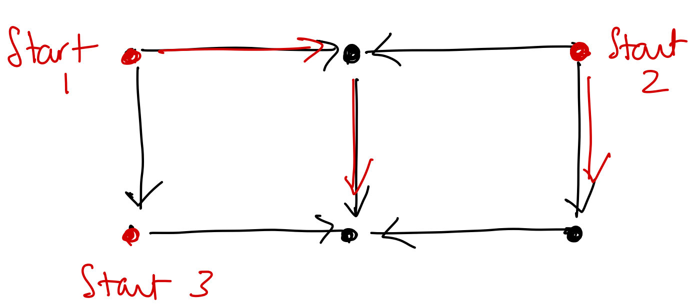


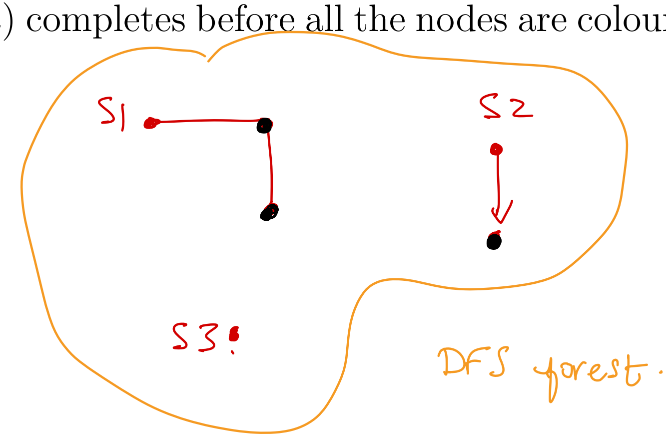
Note that in DFS on undirected graphs, each edge is explored twice.



Consider DFS on possible directed graphs.

- There is no exact notion of a *connected component* (for now).
- The algorithm will restart at a new node if $DFS(u)$ completes before all the nodes are coloured white. The result is a DFS forest.

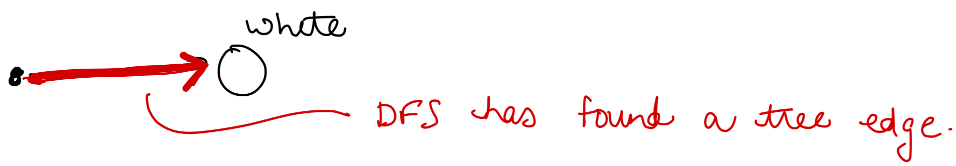




Edge Classification:

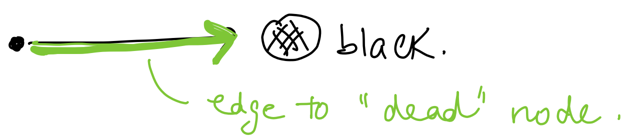
Consider DFS on possible *directed graphs*. There are 3 different types of edge that are *explored* during the execution:

Tree edges:

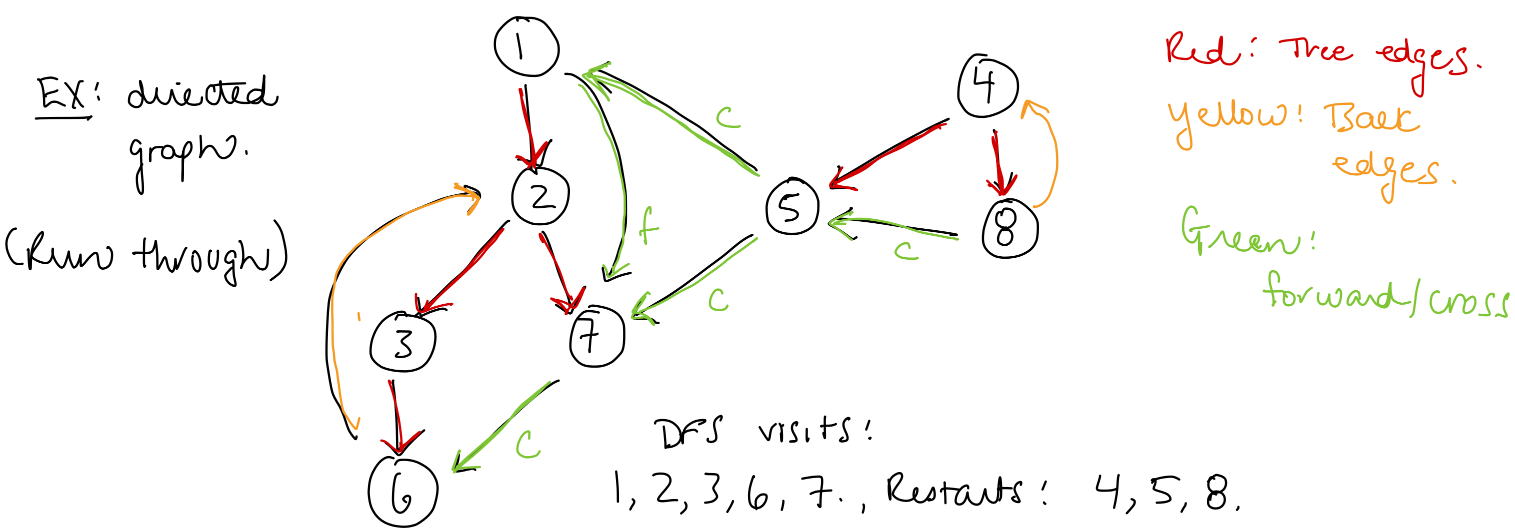


Back edges:

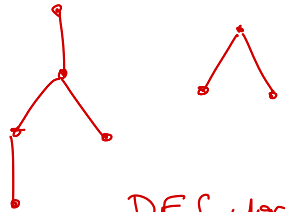
Forward/cross edges:



These edges are shown in the following DFS on a directed graph:



Note that only the **tree edges** represent the DFS forest. The other edges are part of the graph and are classified during execution.

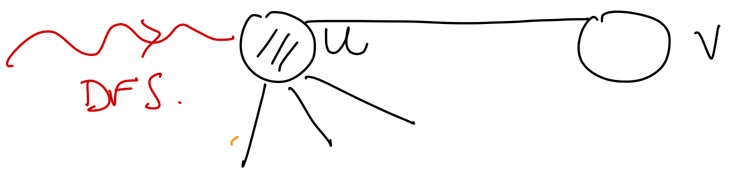


(From above example).

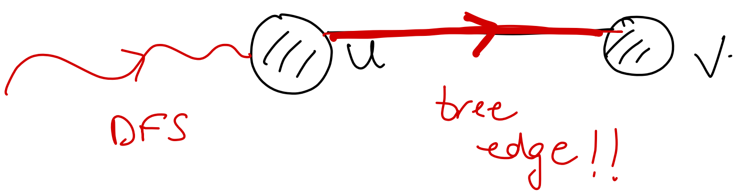
DFS forest.

Theorem: Undirected graphs have no *forward/cross* edges.

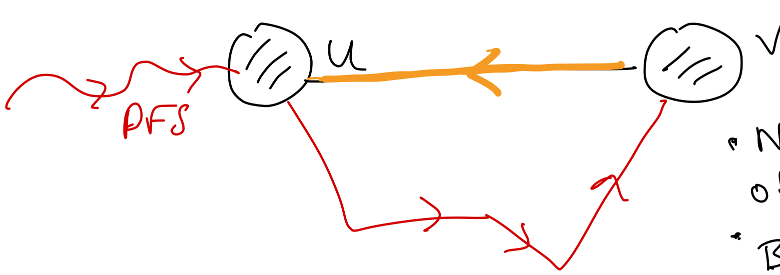
Consider some edge (u, v) in the graph and assume u was discovered first.



CASE 1) DFS could follow edge (u,v):



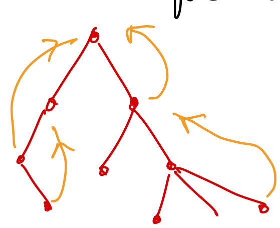
CASE 2) DFS follows some other path that leads to v.



- Now v is a descendent of u
- Before finishing v, DFS will explore edge (v,u).
- Classified: Back edge.

after, v turns black,
then u will turn black ...

Result for undirected graphs:



DFS tree + back edges.