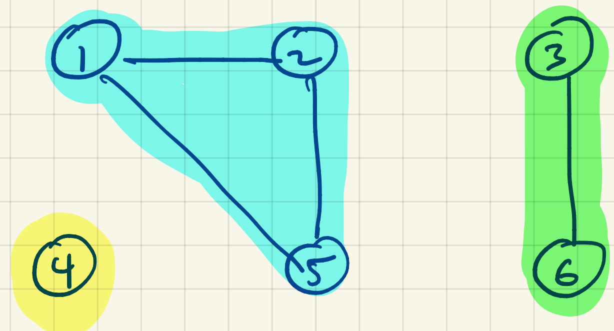
Connected Components

What are connected components of a graph?

Undirected graph

Connected components are the equivalence classes of vertices under the "is reacheable from" relation.

Example:



$\{4\}, \{1,2,5\}, \{3,6\}$

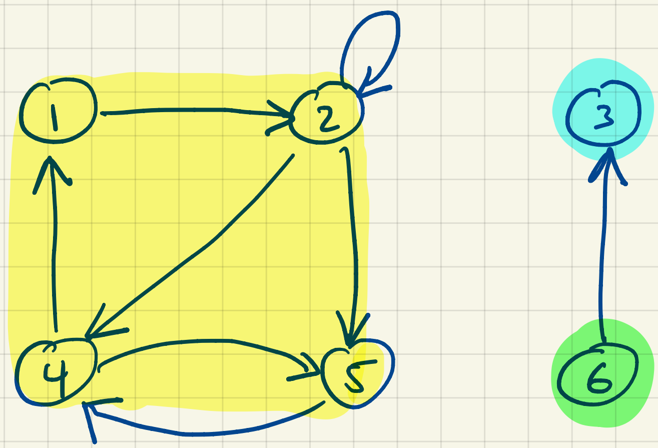
$C_1 = \{1, 2, 5\}$ $\quad C_2 = \{1,2,5\}$

$C_4 = \{4\}$ $\quad C_3 = \{3,6\}$

Directed graphs

Strongly connected components are the equivalence classes of vertices under the "mutually reachable from" relation.

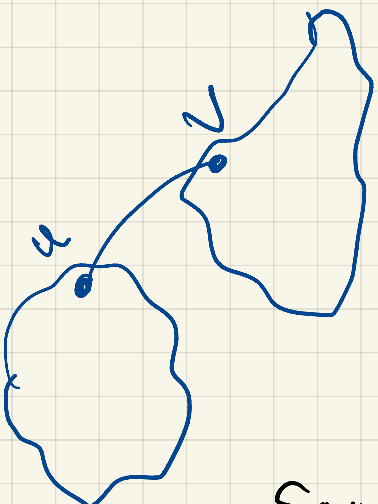
Example:



$\{1,2,4,5\}, \{3\}, \{6\}$

Connected Components in Undirected graphs.

```
Connected_Components (G)
    for each vertex v ∈ V
        do Make_Set (v)
    for each edge (u,v) ∈ E
        do if Find_Set (u) ≠ Find_Set (v)
            then Union (u,v)
```



V Make_Set

E Union

$O(E)$ Find_Set

$\Downarrow$

$O((E+V)\alpha(V))$

```
Same_Component (u,v)
    return Find_Set (u) = Find_Set (v)
```

$O(\alpha(n))$

Recall: A sequence of m Make_Set, Union, and Find_Set operations n of which are Make_Set take $O(m\,\alpha(n))$ time

$\alpha(n)$ is practically a constant ≤ 4.

Connected Components in undirected graphs.

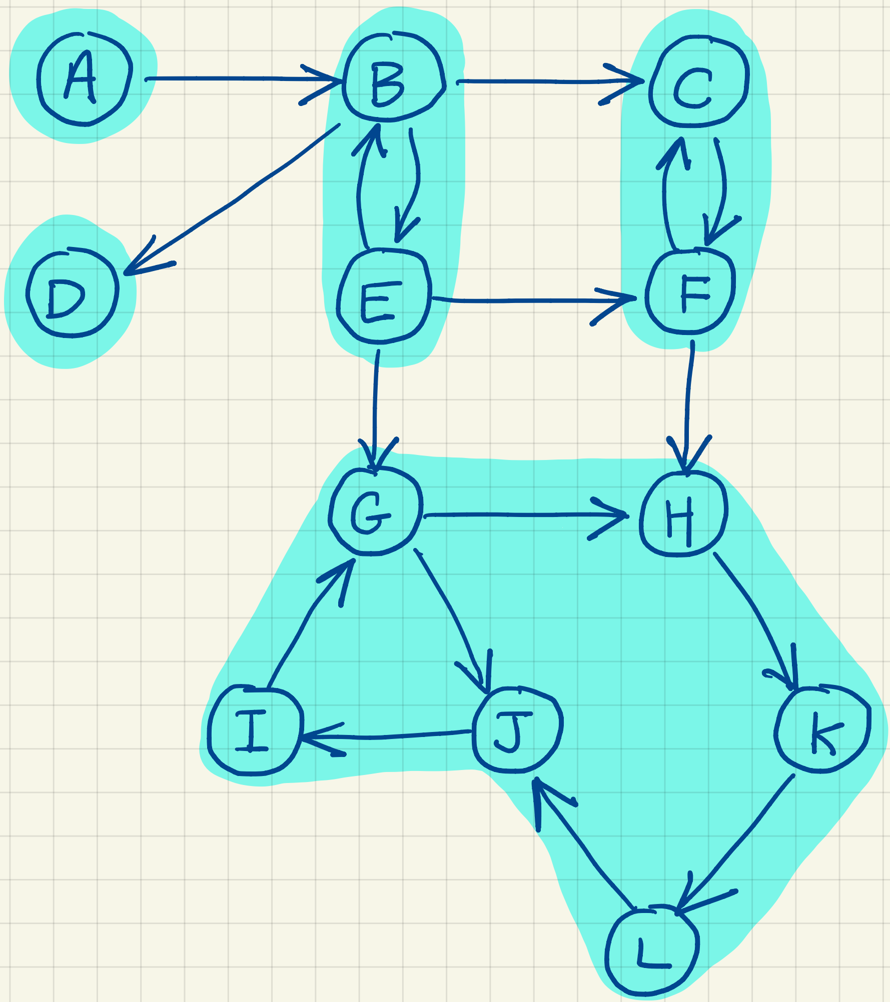
Extension to DFS.

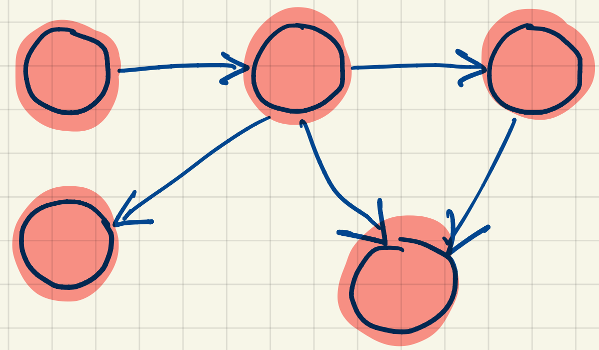
```
DFS (G)
  for each u ∈ V
      do color [u] ← white
  time ← 0                                   component ← 0
  for each u ∈ V
      do if color [u] = white
           then
              DFS-visit (u)
                                             Component ← component + 1
```

```
DFS_visit (u) ........ Called once / vertex
  color [u] ← Gray                           component [u] ← component
  time ← time + 1
  d [u] ← time ................ (Discovered)
  for each v ∈ adj [u]                  }
       do if color [v] = white          }  edges of u
            then DFS-visit (v)          }  explored
  color [u] ← Black
  time ← time + 1
  f [u] ← time .................. (Finished)
```

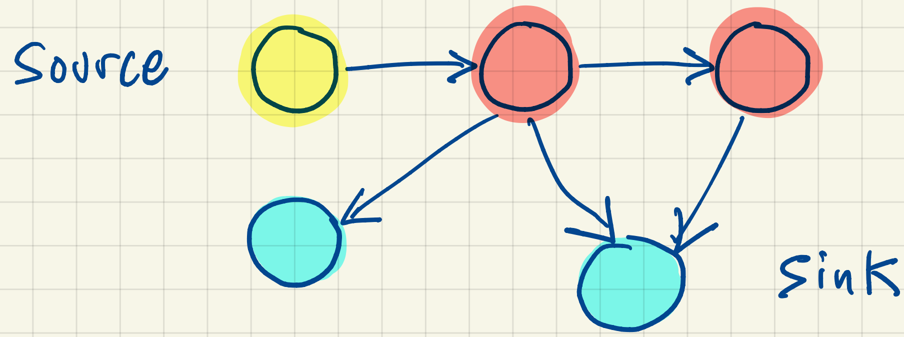
Strongly Connected Components as application of DFS.

Example:





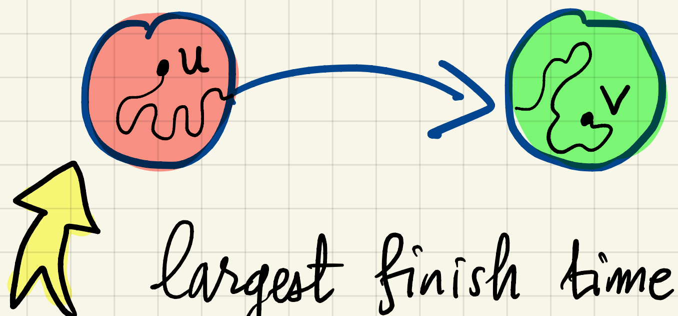
Observation: Every directed graph can be viewed as a DAG of its strongly connected components



DAGs have sources & sinks.

(by topological sort)

Claim 1:



largest finish time is here

If DFS starts with u

$f[u] > f[v]$

Claim 2:

If we run connected components in a sink, we reach only the vertices in that sink.

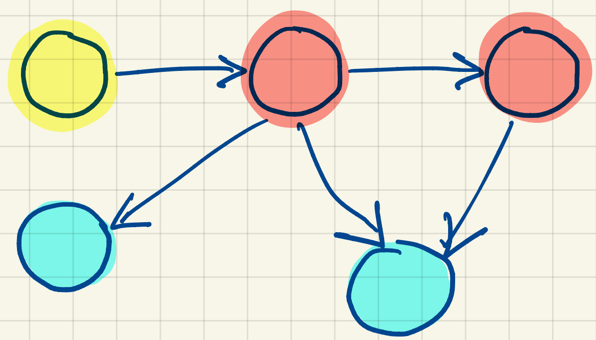
In DFS, vertices in a source component will finish last.

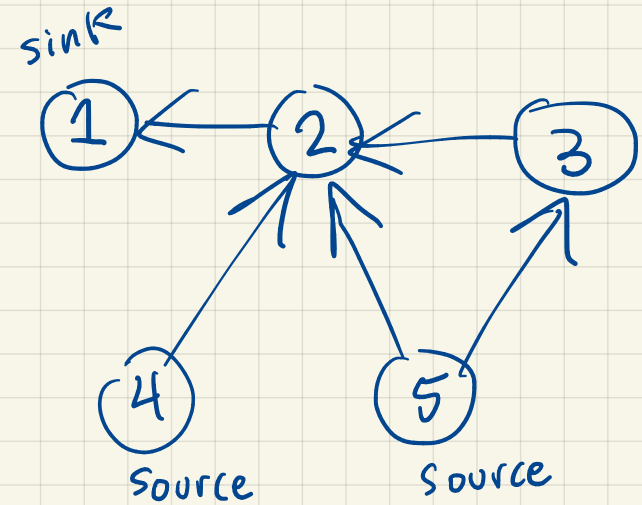
Strongly-Connected Components

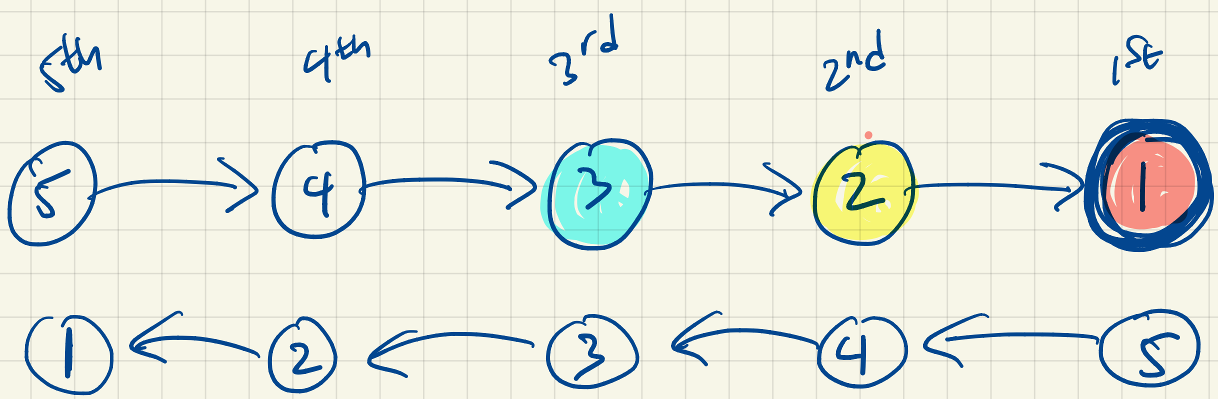
1. Run DFS on G^R (reverse all edges)

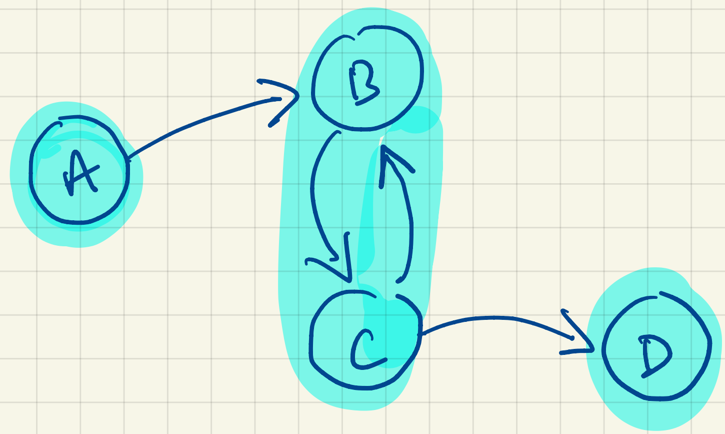
   (now highest $f[u]$ correspond to a sink in G)

2. Run connected components alg. on G based on DFS shown before, with vertices ordered by decreasing $f[u]$

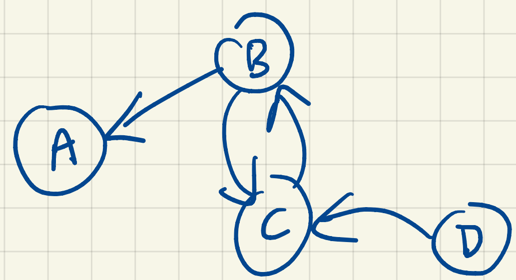



sink
1
2
3
4
5
source
source


5th
4th
3rd
2nd
1st
5
4
3
2
1
1
2
3
4
5



1) Run DFS on:



2) Verify that D has highest finish time.

3) Sort A, B, C, D by decreasing finish time.

Run DFS where main loop goes through A, B, C, D in that order