# Graph Primitives

## An O(m+n) Algorithm for Computing Strong Components

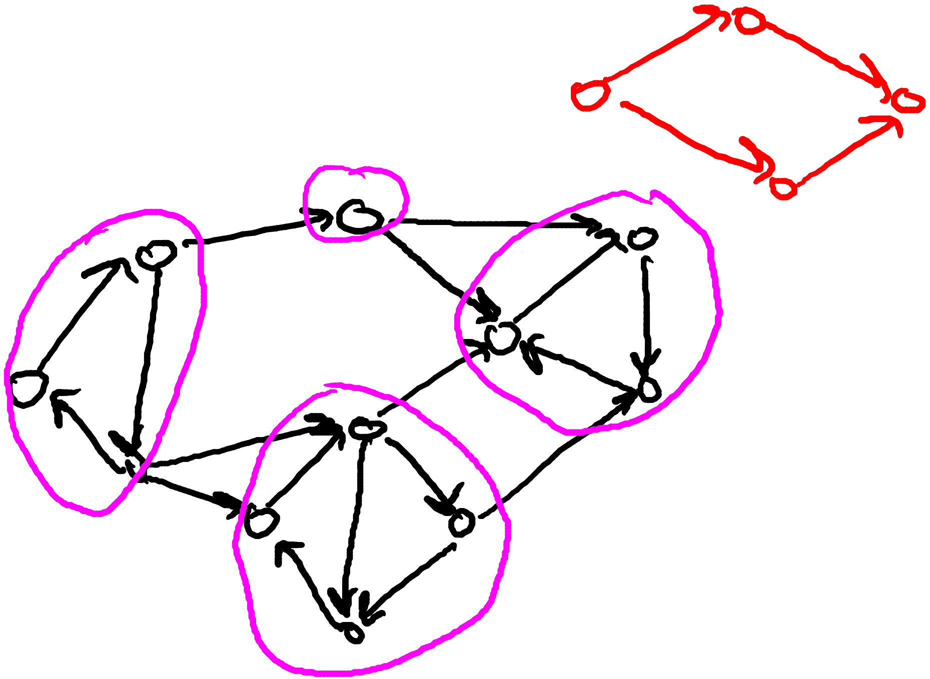
Design and Analysis of Algorithms I

# Strongly Connected Components

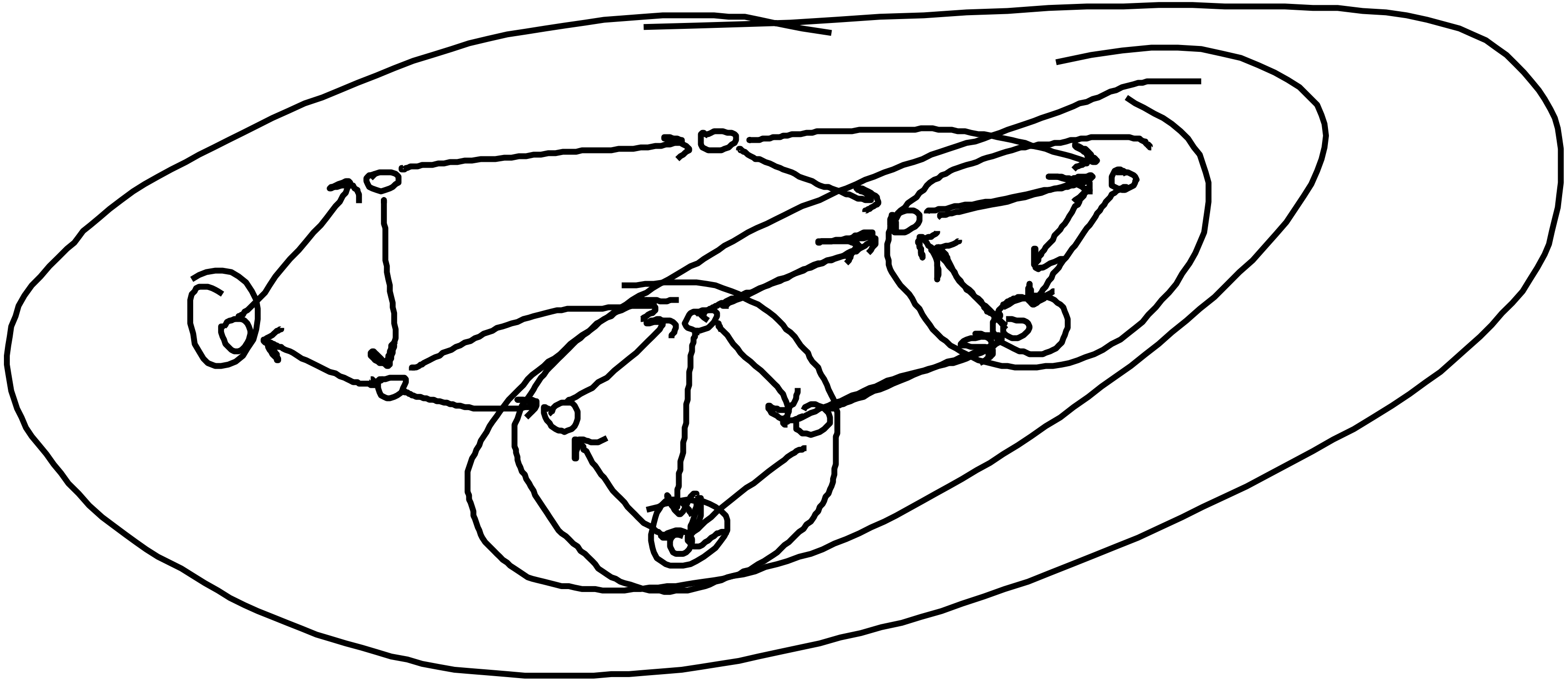
Formal Definition : the strongly connected components (SCCs) of a directed graph G are the equivalence classes of the relation

u<->v <==> there exists a path u->v and a path v->u in G

You check : <-> is an equivalence relation



# Why Depth-First Search?



# Kosaraju's Two-Pass Algorithm

Theorem : can compute SCCs in O(m+n) time.

Algorithm : (given directed graph G)

1. Let Grev = G with all arcs reversed
2. Run DFS-Loop on Grev

   Goal : compute "magical ordering" of nodes

   Let f(v) = "finishing time" of each v in V

1. Run DFS-Loop on G

   Goal : discover the SCCs one-by-one

   processing nodes in decreasing order of finishing times

[ SCCs = nodes with the same "leader" ]

# DFS-Loop

<u>DFS-Loop (graph G)</u>

Global variable t = 0

[# of nodes processed so far]

Global variable s = NULL

[current source vertex]

Assume nodes labeled 1 to n

For i = n down to 1

    if i not yet explored

        s := i

        DFS(G,i)

For finishing times in 1st pass

For leaders in 2nd pass

DFS (graph G, node i)

-- mark i as explored

-- set leader(i) := node s

-- for each arc (i,j) in G :

    -- if j not yet explored

        -- DFS(G,j)

-- t++

-- set f(i) := t

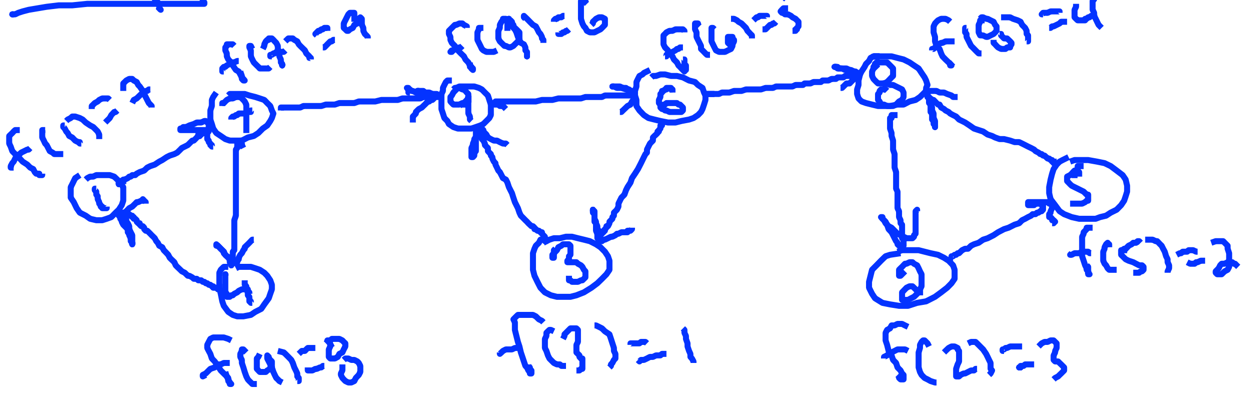
i's finishing time

For rest of DFS-Loop

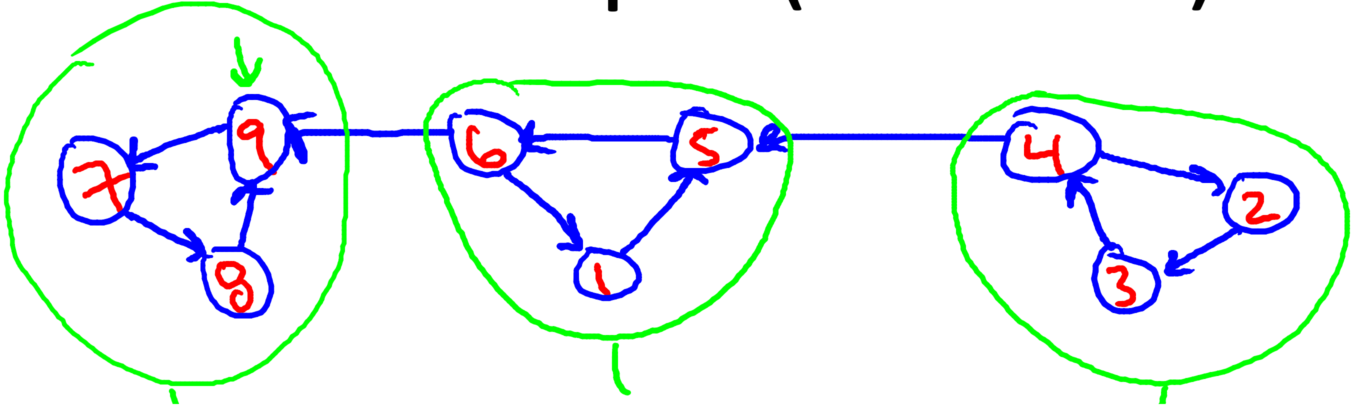
Only one of the following is a possible set of finishing times for the nodes 1,2,3,…,9, respectively, when the DFS-Loop subroutine is executed on the graph below.  Which is it?

Example: [1st DFS-Loop on $G^{rev}$]

- 9,8,7,6,5,4,3,2,1
- 1,7,4,9,6,3,8,2,5
- 1,7,9,6,8,2,5,3,4
- 7,3,1,8,2,5,9,4,6



# Example (2nd Pass)




Leader = 9

Leader = 6

Leader = 4

Running Time : 2*DFS = O(m+n)