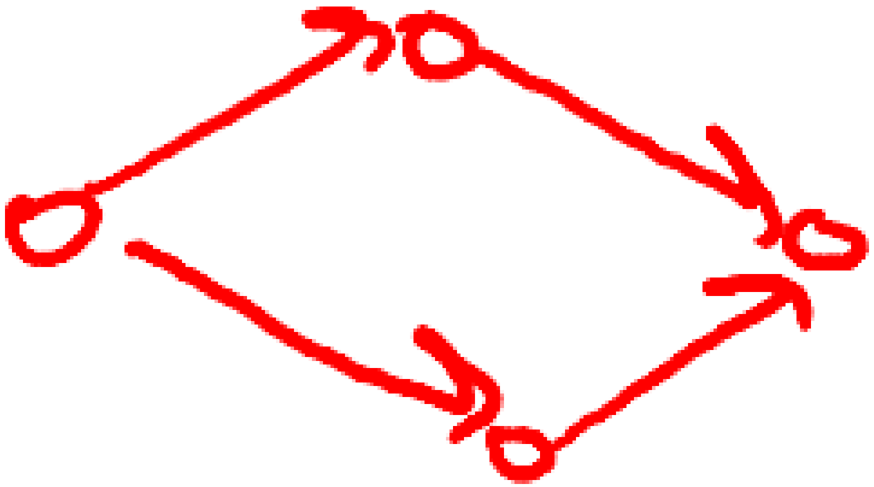
Strongly Connected Components

Application of DFS

Strongly Connected Components (SCC)

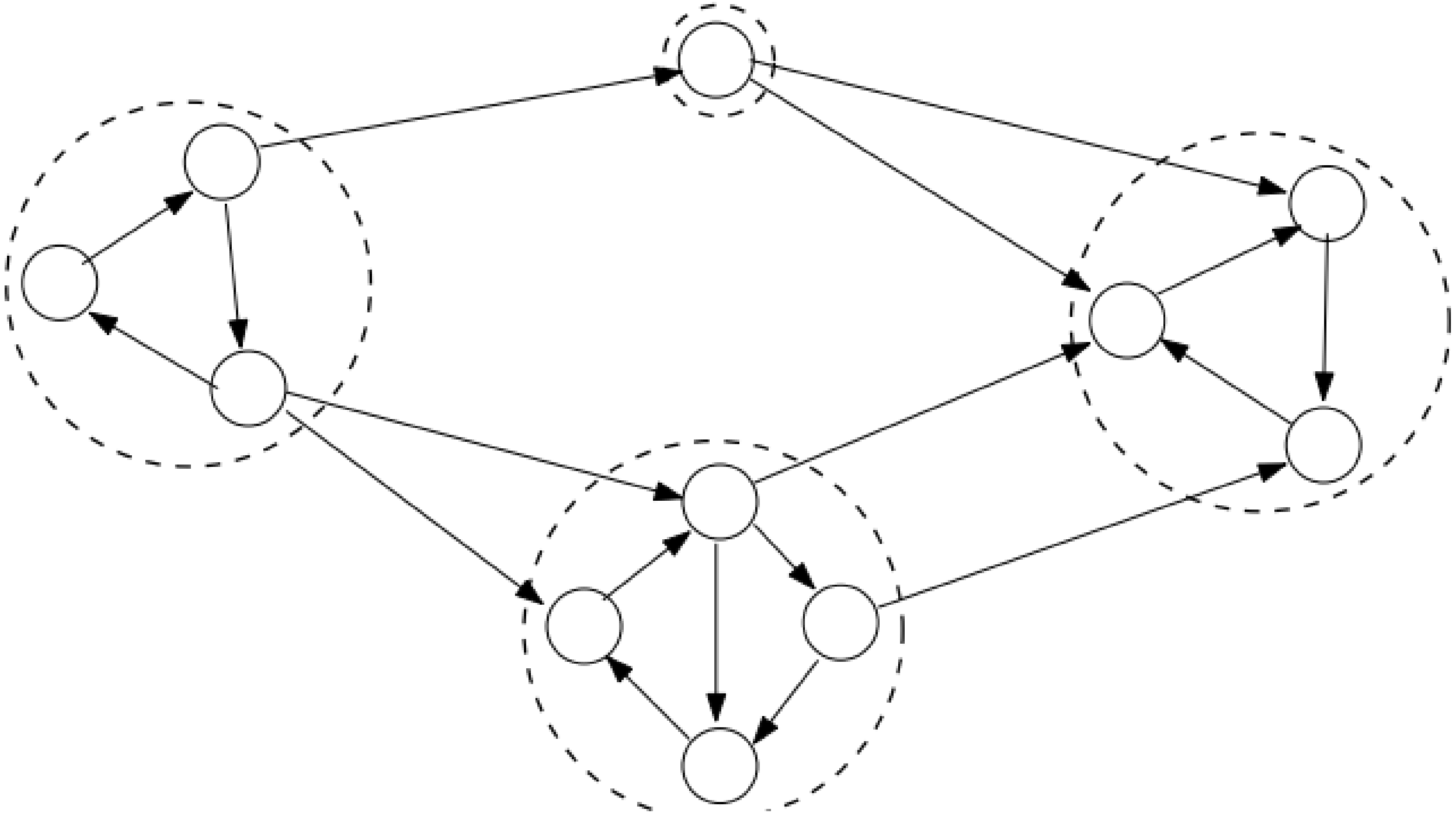
Graph is connected
but not strongly
connected



Strongly Connected Components (SCC)

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



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

DFS-Loop (graph G)

Global variable t = 0

[# of nodes processed so far]

For finishing times in 1^{st} pass

Global variable s = NULL

[current source vertex]

For leaders in 2^{nd} pass

Assume nodes labeled 1 to n

For i = n down to 1

    if i not yet explored

        s := i

        DFS(G,i)

DFS (graph G, node i)

For rest of DFS-Loop

-- mark i as explored

-- set leader(i) := node s

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

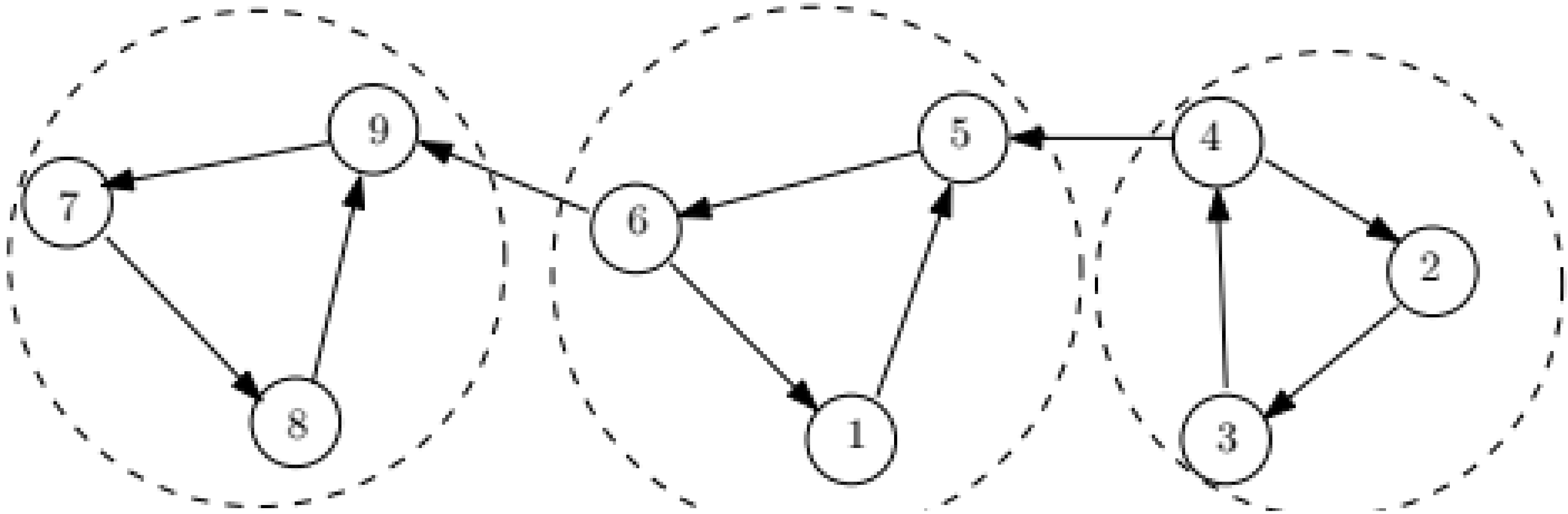
    -- if j not yet explored

        -- DFS(G,j)

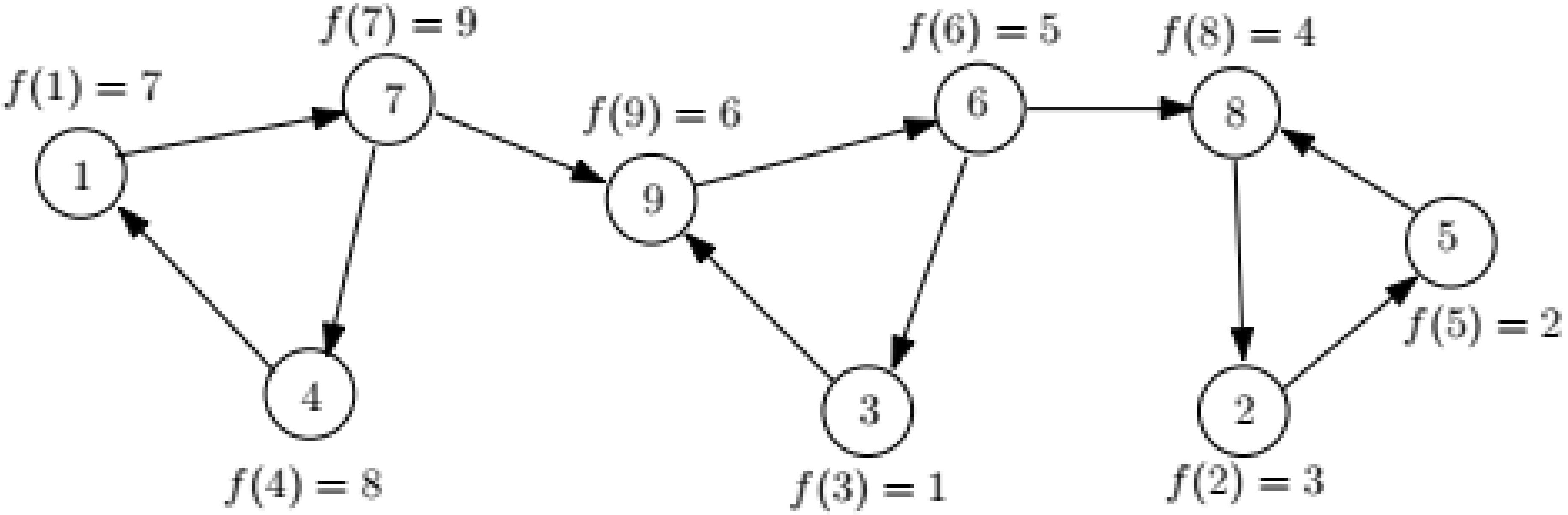
-- t++

-- set f(i) := t

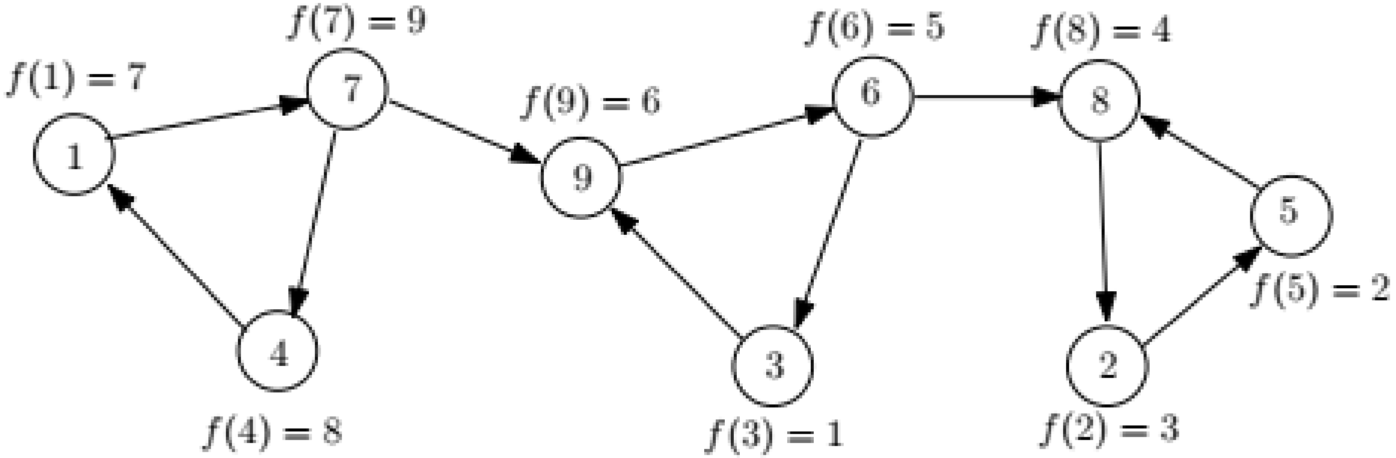
i's finishing time



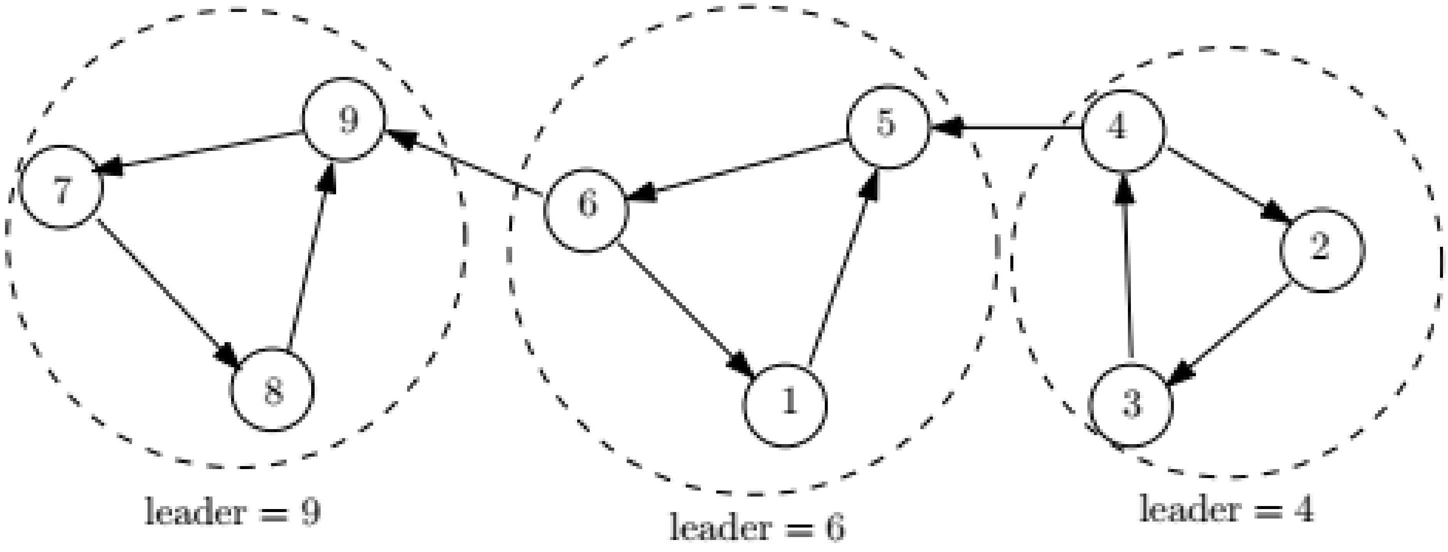
Original Graph



(a) First DFS-Loop on G^{rev}



(a) First DFS-Loop on G^{rev}



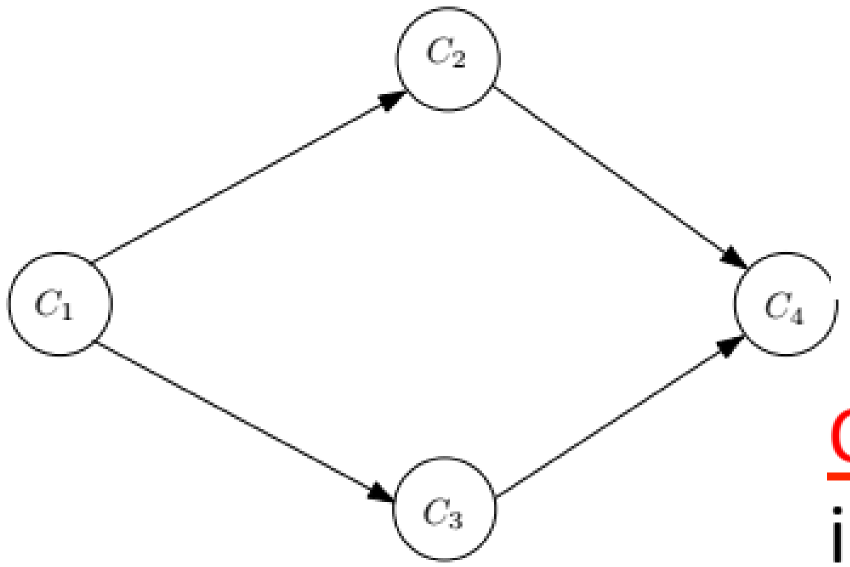
(b) Second DFS-Loop on G

How are the SCC of the original graph G and its reversal G^{rev} related?

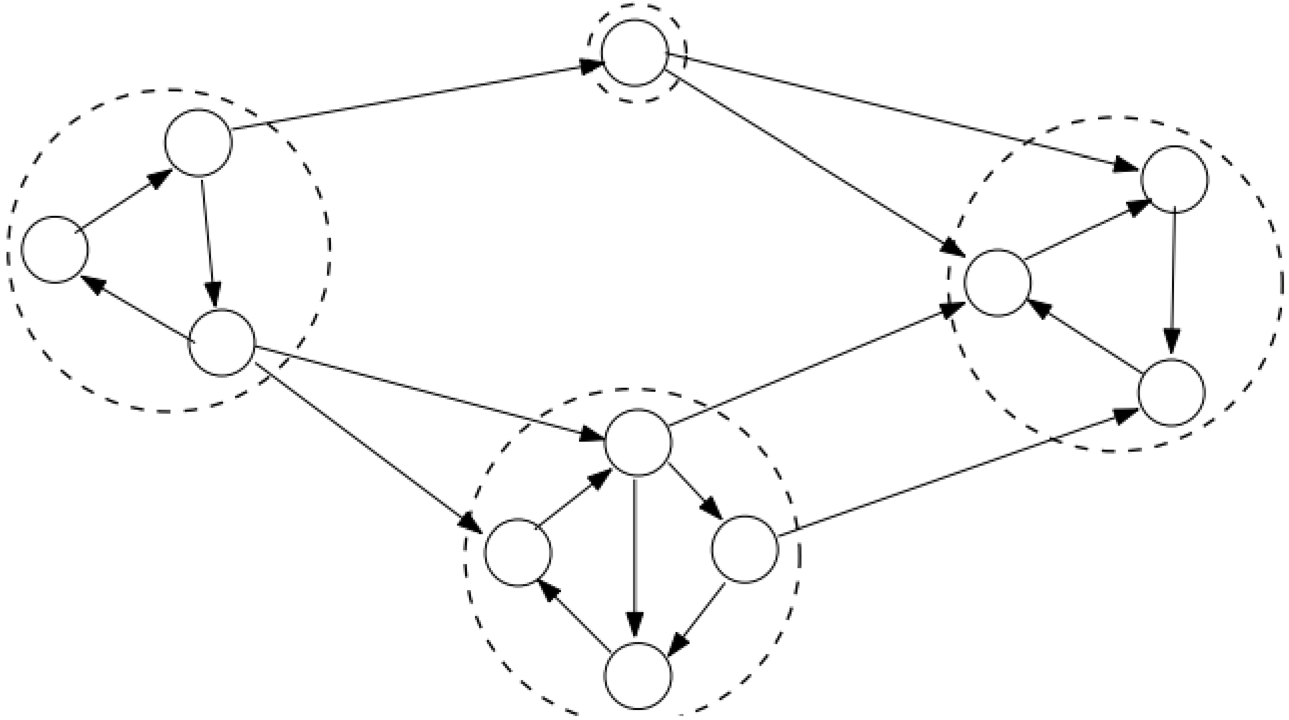
a) In general, they are unrelated.
b) Every SCC of G is contained in an SCC of G^{rev}, but the converse need not hold.
c) Every SCC of G^{rev} is contained in an SCC of G, but the converse need not hold.
d) They are exactly the same.

Directed Acyclic Graph of SCC

Graph of the SCC can never have a cycle. Why ?



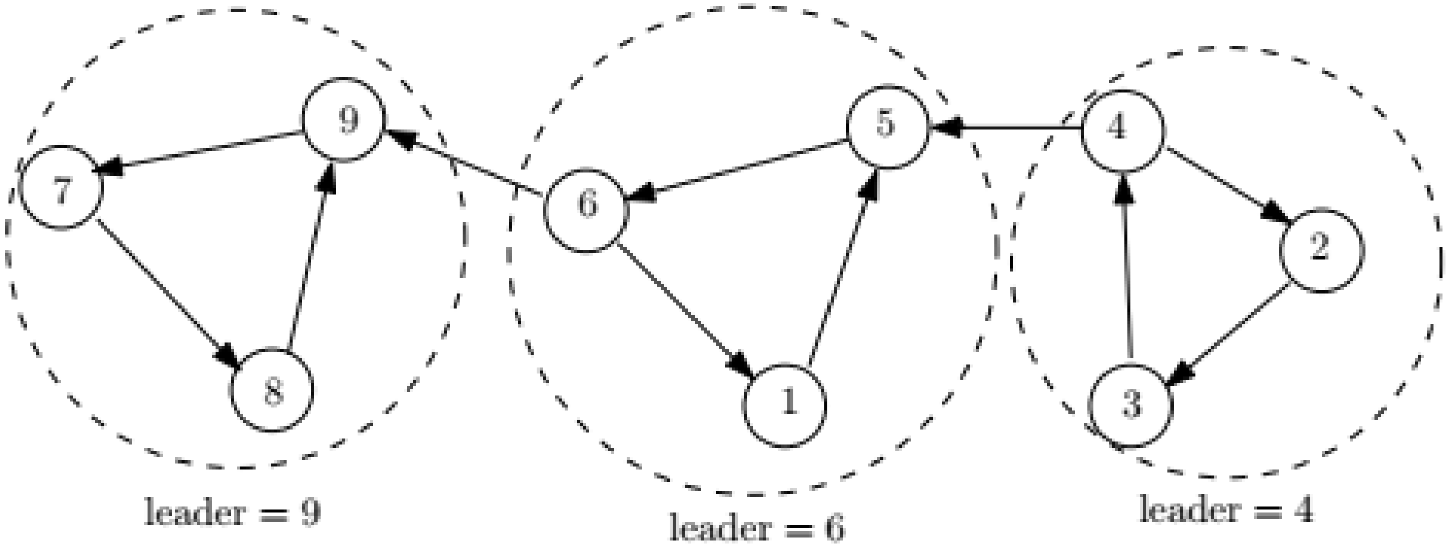
(a) SCC graph for Figure 1

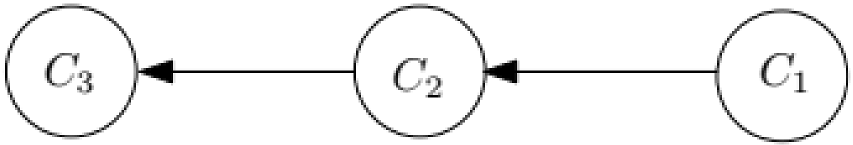


Claim : the SCCs of a directed graph G induce an acyclic "meta-graph":

Directed Acyclic Graph of SCC

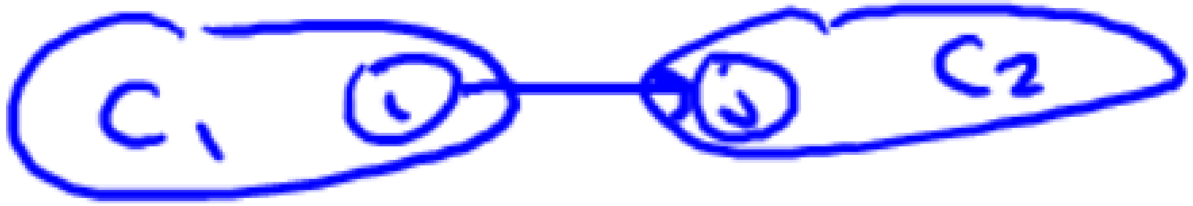
Graph of the SCC can never have a cycle
Reason: All nodes in cycle are reachable from each other so components of cycle should have been part of same component





Correctness Proof of Kosaraju's Algorithm

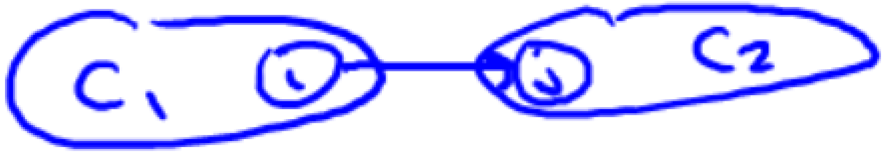
Lemma : consider two "adjacent" SCCs in G:



Let f(v) = finishing times of DFS-Loop in Grev

Then : $max_{v \in C_1} f(v) < max_{v \in C_2} f(v)$

Lemma : consider two "adjacent" SCCs in G:



Let f(v) = finishing times of DFS-Loop in Grev

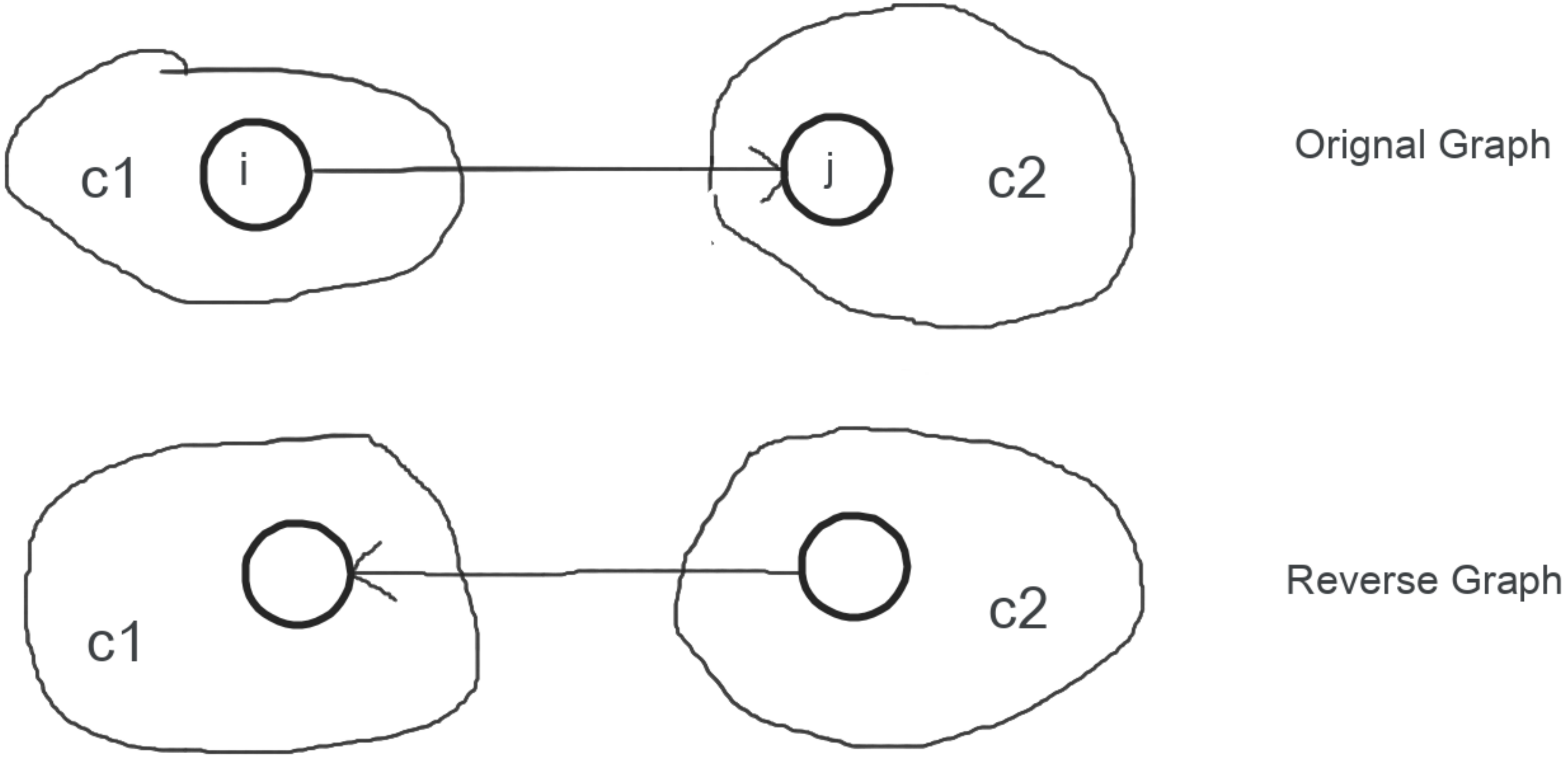
Then : $max_{v \in C_1} f(v) < max_{v \in C_2} f(v)$

Corollary : maximum f-value of G must lie in a "sink SCC"

Max finishing time

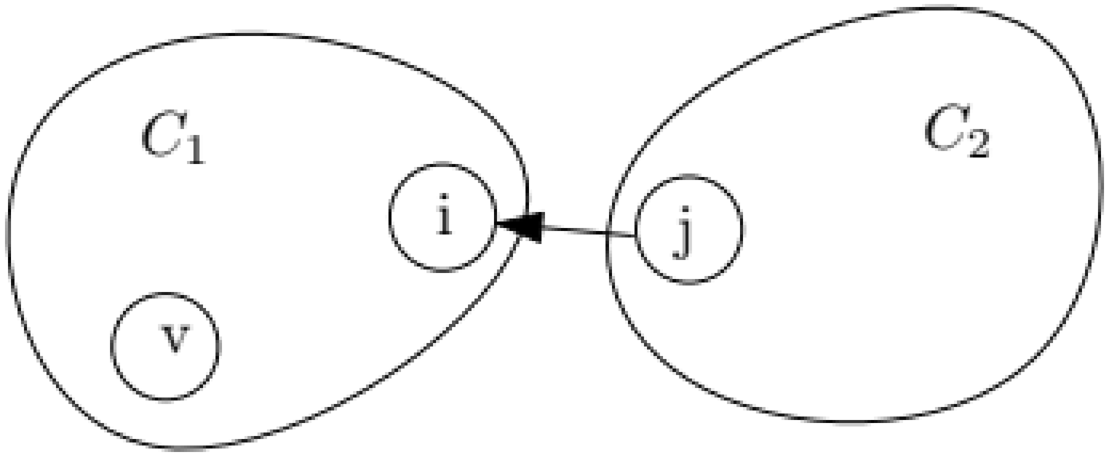
$f_1 < f_2, f_3 < f_4$

Correctness Proof of Kosaraju's Algorithm



$$max_{v \in C_1} f(v) < max_{v \in C_2} f(v)$$

Proof of Correctness



Reverse Graph

Case 1

(a) All f-values in C_1 smaller than in C_2

Let v = 1st node of $C_1 \cup C_2$ reached by 1st pass of DFS-Loop (on Grev)

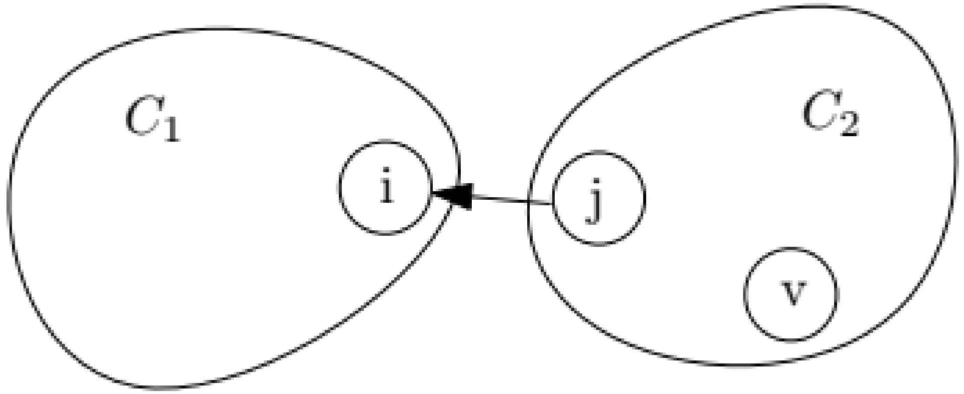
$$max_{v \in C_1} f(v) < max_{v \in C_2} f(v)$$

Case 1 [$v \in C_1$] : all of C_1 explored before C_2 ever reached.

Reason : no paths from C_1 to C_2 (since meta-graph is acyclic)

⇒All f-values in C_1 less than all f-values in C_2

Proof of Correctness



Reverse Graph

(b) v has the largest f-value in $C_1 \cup C_2$

Case 2

Case 2 [$v \in C_2$] : DFS(Grev, v) won't finish until all of $C_1 \cup C_2$ completely explored => f(v) > f(w) for all w in C_1

$$max_{v \in C_1} f(v) < max_{v \in C_2} f(v)$$