**SE 454 Distributed Computing**

**Project**

**Rank-based Vertex Coloring**

***Emin Mert Demirci – 180706023***

***The aim of the project***

This project implements a distributed computing algorithm for rank-based graph coloring using the MPI (Message Passing Interface) library. The algorithm is designed to color nodes in a distributed graph while efficiently communicating with neighboring nodes. This algorithm assumes that the root node is the lowest ranked node, thus terminates when the root node is colored. Each rank/node is trying to find a unique color that is not used by its neighbors.

**High Level Design(FSM)**

**from mpi4py import MPI**

**import numpy as np**

**comm = MPI.COMM\_WORLD**

**rank = comm.Get\_rank()**

**n = comm.Get\_size()**

**# States**

**UNDEC = 0**

**INCOV = 1**

**NOTCOV = 2**

**# Message types**

**ROUND = 0**

**DATA = 1**

**ROVER = 2**

**FINISHED = 3**

**TERMINATE = 4**

**msg = np.array([-1, -1, -1, -1])**

**# Adjacency matrix**

**A = np.array([[0,0,0,1,1,0,0,1],**

**[0,0,0,0,0,1,0,1],**

**[0,0,0,1,1,0,1,0],**

**[1,0,1,0,1,0,1,0],**

**[1,0,1,1,0,1,0,1],**

**[0,1,0,0,1,0,0,1],**

**[0,0,1,1,0,0,0,0],**

**[1,1,0,0,1,1,0,0]], dtype=int)**

**# Spanning tree structure**

**msg = [-1, -1, -1, -1]**

**children = [[3, 4, 7], [], [6], [2], [], [1], [], [5]]**

**parents = [0, 5, 3, 0, 0, 7, 2, 0]**

**# Initialize variables**

**child = children[rank]**

**parent = parents[rank]**

**childs = set(child)**

**neighs, neighs\_rcvd, rover\_rcvd, finish\_rcvd = set(), set(), set(), set()**

**removed\_neighs = set()**

**removed\_childs = set()**

**# Identify neighbors**

**for i in range(n):**

**if A[rank,i] == 1:**

**neighs.add(i)**

**currneighs = neighs.copy()**

**currchilds = childs.copy()**

**round\_num = 0**

**max\_count = n - 1**

**state = UNDEC**

**finished\_childs, finished\_all, terminated = False, False, False**

**# Main algorithm loop**

**while not terminated:**

**round\_over, neighs\_over, childs\_over = False, False, False**

**neighs\_sent = False**

**rover\_rcvd.clear()**

**neighs\_rcvd.clear()**

**removed\_neighs.clear()**

**removed\_childs.clear()**

**# Root node logic**

**if rank == 0:**

**msg = [rank, ROUND, round\_num, -1]**

**for child in currchilds:**

**comm.send(msg, dest=child, tag=ROUND)**

**if state == UNDEC:**

**msg[1], msg[3] = DATA, state**

**for node in currneighs:**

**comm.send(msg, dest=node, tag=DATA)**

**neighs\_sent = True**

**else:**

**neighs\_over = True**

**neighs\_sent = True**

**# Main communication loop**

**while not round\_over:**

**msg = comm.recv(source=MPI.ANY\_SOURCE, tag=MPI.ANY\_TAG)**

**sender, typ, roun, ne\_state = msg[0], msg[1], msg[2], msg[3]**

**msg[0] = rank**

**if typ == ROUND:**

**if len(currchilds) != 0:**

**for child in currchilds:**

**comm.send(msg, dest=child, tag=ROUND)**

**else:**

**childs\_over = True**

**if state != UNDEC:**

**neighs\_over = True**

**neighs\_sent = True**

**if len(childs) == 0:**

**childs\_over = True**

**msg[1] = ROVER**

**comm.send(msg, dest=parent, tag=ROVER)**

**round\_over = True**

**else:**

**neighs\_sent, neighs\_over = True, True**

**else:**

**if rank > max(currneighs):**

**state = INCOV**

**round\_saved = max\_count - round\_num**

**if len(currchilds) == 0:**

**childs\_over = True**

**finished\_all = True**

**msg[1] = FINISHED**

**comm.send(msg, dest=parent, tag=FINISHED)**

**msg[1], msg[3] = DATA, state**

**for node in currneighs:**

**comm.send(msg, dest=node, tag=DATA)**

**neighs\_sent = True**

**if neighs\_over and childs\_over:**

**if rank != 0:**

**msg[1] = ROVER**

**comm.send(msg, dest=parent, tag=ROVER)**

**round\_over = True**

**elif typ == DATA:**

**neighs\_rcvd.add(sender)**

**if ne\_state == INCOV or ne\_state == NOTCOV:**

**if sender in currneighs:**

**removed\_neighs.add(sender)**

**if neighs\_rcvd == currneighs:**

**neighs\_over = True**

**if neighs\_sent:**

**if len(childs) == 0 or childs\_over:**

**if rank != 0:**

**msg[1] = ROVER**

**comm.send(msg, dest=parent, tag=ROVER)**

**round\_over = True**

**elif typ == ROVER:**

**rover\_rcvd.add(sender)**

**if rover\_rcvd == currchilds:**

**childs\_over = True**

**if neighs\_over and neighs\_sent:**

**round\_over = True**

**if rank != 0:**

**msg[1] = ROVER**

**comm.send(msg, dest=parent, tag=ROVER)**

**elif typ == FINISHED:**

**finish\_rcvd.add(sender)**

**removed\_childs.add(sender)**

**if finish\_rcvd == childs:**

**finished\_childs = True**

**if state != UNDEC:**

**finished\_all = True**

**if rank != 0:**

**msg[1] = FINISHED**

**comm.send(msg, dest=parent, tag=FINISHED)**

**else:**

**msg[1] = TERMINATE**

**for child in childs:**

**comm.send(msg, dest=child, tag=TERMINATE)**

**round\_over = True**

**terminated = True**

**elif typ == TERMINATE:**

**if len(childs) != 0:**

**for child in childs:**

**comm.send(msg, dest=child, tag=TERMINATE)**

**terminated = True**

**round\_over = True**

**round\_num += 1**

**currneighs -= removed\_neighs**

**currchilds -= removed\_childs**

**if len(currneighs) == 0 and state == UNDEC:**

**state = NOTCOV**

**round\_saved = max\_count - round\_num**

**if len(currchilds) == 0:**

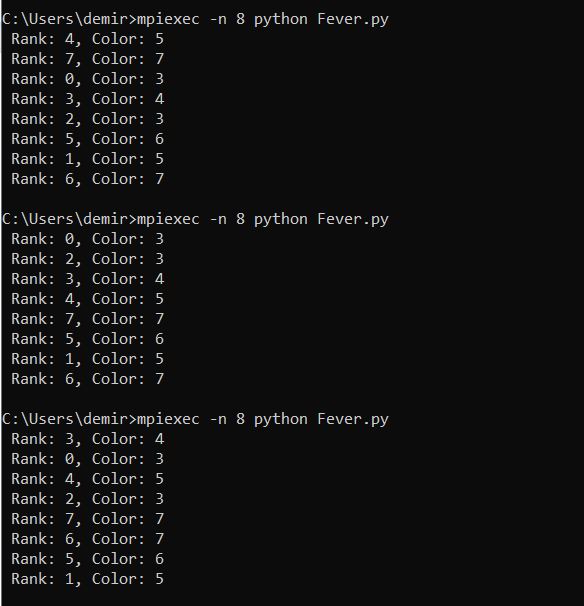
**finished\_all = True**

**msg[1] = FINISHED**

**comm.send(msg, dest=parent, tag=FINISHED)**

**print(" Rank: {}, Color: {}".format(rank, round\_saved))**

**Output:**

****

**Pseudocode:  
  
1: int rank ← MPI\_Comm\_rank()**

**2: int n ← MPI\_Comm\_size()**

**3: enum state {UNDEC, INCOV, NOTCOV}**

**4: enum message\_type {ROUND, DATA, ROVER, FINISHED, TERMINATE}**

**5: int state ← UNDEC**

**6: array of int msg ← [⊥, ⊥, ⊥, ⊥]**

**7: matrix int A[n][n] ← ... // Adjacency matrix**

**8: list of list of int children ← [[3, 4, 7], [], [6], [2], [], [1], [], [5]]**

**9: list of int parents ← [0, 5, 3, 0, 0, 7, 2, 0]**

**10: set of int childs ← children[rank]**

**11: set of int neighs ← ∅**

**12: set of int neighs\_rcvd, rover\_rcvd, finish\_rcvd ← ∅**

**13: int round\_num ← 0**

**14: boolean finished\_childs ← false, finished\_all ← false, terminated ← false**

**15:**

**16: for i ← 0 to n-1 do**

**17: if A[rank][i] = 1 then**

**18: neighs.add(i)**

**19: end if**

**20: end for**

**21: set of int currneighs ← neighs.copy()**

**22: set of int currchilds ← childs.copy()**

**23: int round\_saved**

**24:**

**25: while ¬terminated do**

**26: boolean round\_over ← false, neighs\_over ← false, childs\_over ← false**

**27: boolean neighs\_sent ← false**

**28: clear rover\_rcvd, neighs\_rcvd**

**29:**

**30: if rank = 0 then**

**31: prepare and send ROUND message to children**

**32: if state = UNDEC then**

**33: prepare and send DATA message to neighbors**

**34: set neighs\_sent to true**

**35: else**

**36: set neighs\_over to true and neighs\_sent to true**

**37: end if**

**38:**

**39: while ¬round\_over do**

**40: msg ← receive message**

**41: handle ROUND message**

**42: handle DATA message**

**43: handle ROVER message**

**44: handle FINISHED message**

**45: handle TERMINATE message**

**46: update states and perform transitions based on received messages**

**47: end while**

**48:**

**49: increment round\_num**

**50: update current neighbors and children**

**51:**

**52: if no neighbors and state = UNDEC then**

**53: set state to NOTCOV**

**54: if no children then**

**55: signal completion to parent**

**56: end if**

**57: end if**

**58: end while**

**59: output rank and coloring information**

**End Algorithm**

**Another Way:  
It’s starts from color: 0**

**from mpi4py import MPI**

**import numpy as np**

**comm = MPI.COMM\_WORLD**

**rank = comm.Get\_rank()**

**n = comm.Get\_size()**

**ROUND = 0**

**COLOR = 1**

**ROVER = 2**

**FIN = 3**

**msg = np.array([-1, -1, -1, -1])**

**children = [[3, 4, 7], [], [6], [2], [], [1], [], [5]]**

**parents = [0, 5, 3, 0, 0, 7, 2, 0]**

**finished, round\_end = set(), set()**

**childs = children[rank]**

**parent = parents[rank]**

**childs = set(childs)**

**neighs, received = set(), set()**

**colors = {}**

**A = np.array([[0,0,0,1,1,0,0,1],**

**[0,0,0,0,0,1,0,1],**

**[0,0,0,1,1,0,1,0],**

**[1,0,1,0,1,0,1,0],**

**[1,0,1,1,0,1,0,1],**

**[0,1,0,0,1,0,0,1],**

**[0,0,1,1,0,0,0,0],**

**[1,1,0,0,1,1,0,0]], dtype=int)**

**neighbors = A[rank,:]**

**s = []**

**for i in range (0,n):**

**if neighbors[i] == 1:**

**neighs.add(i)**

**colors.update([(i,-1)])**

**count = n - 1**

**round\_num = 0**

**color = -1**

**changed = False**

**if 0 in neighs:**

**colors[0] = 0**

**if rank == 0:**

**color = 0**

**while count > 0:**

**neighs\_over, childs\_end, round\_over = False, False, False**

**round\_end.clear()**

**if rank == 0:**

**round\_num = round\_num + 1**

**msg[0], msg[1], msg[2] = rank, ROUND, round\_num**

**for child in childs:**

**comm.send(msg, dest=child, tag=ROUND)**

**while not round\_over:**

**msg = comm.recv(source=MPI.ANY\_SOURCE, tag=MPI.ANY\_TAG)**

**sender, typ, roun, neigh\_color=msg[0], msg[1], msg[2], msg[3]**

**msg[0] = rank**

**if typ == ROUND:**

**if len(childs) != 0:**

**for child in childs:**

**comm.send(msg, dest=child, tag=ROUND)**

**else:**

**childs\_end = True**

**if rank == roun:**

**for key,value in colors.items():#find an unused color**

**s.append(value)**

**s.sort()**

**found = False**

**for i in range(1,len(s)):**

**if s[i] - s[i-1] > 1:**

**color = s[i-1] + 1**

**found = True**

**break**

**if not found:**

**color = max(s) + 1**

**colors[rank] = color**

**msg[1] = COLOR**

**msg[3] = color**

**for node in neighs:**

**comm.send(msg, dest=node, tag=COLOR)**

**changed = True**

**if len(childs) == 0 and roun not in neighs:**

**msg[1] = ROVER**

**comm.send(msg, dest=parent, tag=ROVER)**

**round\_over = True**

**elif typ == COLOR:**

**colors[sender] = neigh\_color**

**changed = True**

**if childs\_end:**

**if rank != 0:**

**msg[1] = ROVER**

**comm.send(msg, dest=parent, tag=ROVER)**

**round\_over = True**

**elif typ == ROVER:**

**round\_end.add(sender)**

**if round\_end == childs:**

**childs\_end = True**

**if roun == rank:**

**if changed:**

**round\_over = True**

**elif roun in neighs:**

**if changed:**

**round\_over = True**

**else:**

**round\_over = True**

**if rank != 0:**

**comm.send(msg, dest=parent, tag=ROVER)**

**count = count - 1**

**print("rank: {}, color: {}".format(rank,color))**

**Pseudocode:**

**1: int rank ← MPI\_Comm\_rank(), n ← MPI\_Comm\_size()**

**2: int color ← -1, round\_num ← 0**

**3: boolean changed ← false, round\_over ← false, childs\_end ← false**

**4: set of int children ← [[3, 4, 7], [], [6], [2], [], [1], [], [5]]**

**5: int parent ← [0, 5, 3, 0, 0, 7, 2, 0]**

**6: set of int childs ← children[rank], neighs ← ∅, finished ← ∅, round\_end ← ∅**

**7: array of int msg ← [-1, -1, -1, -1]**

**8: dictionary of int colors ← {}**

**9: int A[n][n] ← ... // adjacency matrix**

**10: int count ← n - 1**

**11: message types ROUND, COLOR, ROVER, FIN**

**12:**

**13: for int i ← 0 to n do**

**14: if A[rank][i] = 1 then**

**15: neighs ← neighs ∪ {i}**

**16: colors[i] ← -1**

**17: end if**

**18: end for**

**19:**

**20: if rank = 0 then**

**21: color ← 0**

**22: end if**

**23:**

**24: while count > 0 do**

**25: if rank = 0 then**

**26: round\_num ← round\_num + 1**

**27: msg ← [rank, ROUND, round\_num, -1]**

**28: for each child in childs do**

**29: MPI\_Send msg to child with tag ROUND**

**30: end for**

**31: end if**

**32:**

**33: while ¬round\_over do**

**34: msg ← MPI\_Recv from MPI\_ANY\_SOURCE, MPI\_ANY\_TAG**

**35: int sender, typ, roun, neigh\_color ← msg[0], msg[1], msg[2], msg[3]**

**36: msg[0] ← rank**

**37: case msg[1] of**

**38: ROUND:**

**39: // Process ROUND message**

**40: COLOR:**

**41: // Process COLOR message**

**42: ROVER:**

**43: // Process ROVER message**

**44: end case**

**45: end while**

**46:**

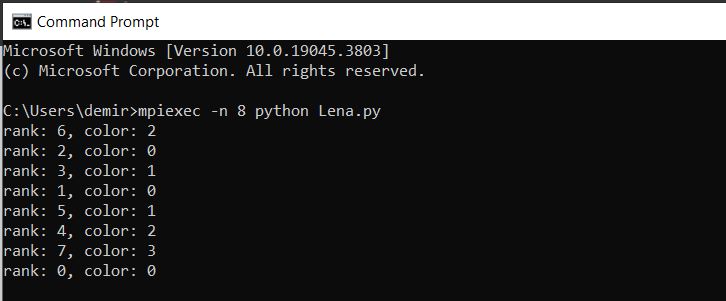
**47: count ← count - 1**

**48: end while**

**49: MPI\_Finalize()**

**50: print "rank: ", rank, " color: ", color**

**Output:**

****

**FSM Design  
  
from mpi4py import MPI**

**import numpy as np**

**comm = MPI.COMM\_WORLD**

**rank = comm.Get\_rank()**

**n = comm.Get\_size()**

**#states**

**UNDEC = 0**

**INCOV = 1**

**NOTCOV = 2**

**# message types**

**ROUND = 0**

**DATA = 1**

**ROVER = 2**

**FINISHED = 3**

**TERMINATE = 4**

**# sender, type, round number, sender state**

**msg = np.array([-1, -1, -1, -1])**

**A = np.array([[0,0,0,1,1,0,0,1],**

**[0,0,0,0,0,1,0,1],**

**[0,0,0,1,1,0,1,0],**

**[1,0,1,0,1,0,1,0],**

**[1,0,1,1,0,1,0,1],**

**[0,1,0,0,1,0,0,1],**

**[0,0,1,1,0,0,0,0],**

**[1,1,0,0,1,1,0,0]], dtype=int)**

**#sender, type, round number, sender state**

**msg = [-1, -1, -1, -1]**

**children = [[3, 4, 7], [], [6], [2], [], [1], [], [5]]**

**parents = [0, 5, 3, 0, 0, 7, 2, 0]**

**#children = [[3, 4, 7], [], [], [6], [], [1], [2], [5]]**

**#parents = [0, 5, 6, 0, 0, 7, 3, 0]**

**#children = [[3, 4, 7], [], [], [2,6], [], [], [], [1,5]]**

**#parents = [0, 7, 3, 0, 0, 7, 3, 0]**

**child = children[rank]**

**parent = parents[rank]**

**childs = set(child)**

**neighs, neighs\_rcvd, rover\_rcvd, finish\_rcvd = set(), set(), set(), set()**

**removed\_neighs = set()**

**removed\_childs = set()**

**states = ['UNDEC', 'INCOV', 'NOTCOV']**

**for i in range (0,n): #identify neighbors**

**if A[rank,i] == 1:**

**neighs.add(i)**

**currneighs = neighs.copy()**

**currchilds = childs.copy()**

**round\_num = 0**

**max\_count = 4**

**state = UNDEC**

**finished\_childs, finished\_all, terminated = False, False, False**

**while not terminated:**

**round\_over, neighs\_over, childs\_over = False, False, False**

**neighs\_sent = False**

**rover\_rcvd.clear()**

**neighs\_rcvd.clear()**

**removed\_neighs.clear()**

**removed\_childs.clear()**

**if rank == 0:**

**msg[0], msg[1], msg[2] = rank, ROUND, round\_num**

**for child in currchilds:**

**comm.send(msg, dest=child, tag=ROUND)**

**if state==UNDEC:**

**msg[1], msg[3] = DATA, state**

**for node in currneighs:**

**comm.send(msg, dest=node, tag=DATA)**

**neighs\_sent = True**

**else:**

**neighs\_over = True**

**neighs\_sent = True**

**while not round\_over:**

**msg = comm.recv(source=MPI.ANY\_SOURCE, tag=MPI.ANY\_TAG)**

**sender, typ, roun, ne\_state = msg[0], msg[1], msg[2], msg[3]**

**msg[0] = rank**

**if typ == ROUND: #active node ROUND recieved**

**if len(currchilds) != 0: #intermediate node**

**for child in currchilds:**

**comm.send(msg, dest=child, tag=ROUND)**

**else: #leaf node**

**childs\_over = True**

**if state != UNDEC: #node already determined**

**neighs\_over = True**

**neighs\_sent = True**

**if len(childs)==0:**

**childs\_over = True**

**msg[1] = ROVER**

**comm.send(msg, dest=parent, tag=ROVER)**

**round\_over = True**

**else:**

**neighs\_sent, neighs\_over = True, True**

**else: #node undetermined**

**if rank > max(currneighs):**

**state = INCOV**

**round\_saved = roun**

**if len(currchilds) == 0:**

**childs\_over = True**

**finished\_all = True**

**msg[1] = FINISHED**

**comm.send(msg,dest=parent,tag=FINISHED)**

**msg[1], msg[3] = DATA, state**

**for node in currneighs:**

**comm.send(msg, dest=node, tag=DATA)**

**neighs\_sent = True**

**if neighs\_over and childs\_over:**

**if rank != 0:**

**msg[1] = ROVER**

**comm.send(msg, dest=parent, tag=ROVER)**

**round\_over = True**

**elif typ == DATA: #DATA received**

**neighs\_rcvd.add(sender)**

**if ne\_state == INCOV or ne\_state == NOTCOV:**

**if sender in currneighs:**

**removed\_neighs.add(sender)**

**if neighs\_rcvd == currneighs:**

**neighs\_over = True**

**if neighs\_sent:**

**# childs finished?**

**if len(childs)==0 or childs\_over:**

**if rank != 0:**

**msg[1] = ROVER**

**comm.send(msg, dest=parent, tag=ROVER)**

**round\_over = True**

**elif typ == ROVER: # ROVER received**

**rover\_rcvd.add(sender)**

**if rover\_rcvd == currchilds:**

**childs\_over = True**

**if neighs\_over and neighs\_sent: #neighs finished**

**round\_over = True**

**if rank != 0:**

**msg[1] = ROVER**

**comm.send(msg,dest=parent,tag=ROVER)**

**elif typ == FINISHED: # a child finished**

**finish\_rcvd.add(sender)**

**removed\_childs.add(sender)**

**if finish\_rcvd == childs:**

**finished\_childs = True**

**if state != UNDEC:**

**finished\_all = True**

**if rank != 0:**

**msg[1] = FINISHED**

**comm.send(msg,dest=parent,tag=FINISHED)**

**else:**

**msg[1] = TERMINATE**

**for child in childs:**

**comm.send(msg, dest=child, tag=TERMINATE)**

**round\_over = True**

**terminated = True**

**else: #TERMINATE received**

**if len(childs) != 0:**

**for child in childs:**

**comm.send(msg, dest=child, tag=TERMINATE)**

**terminated = True**

**round\_over = True**

**round\_num = round\_num + 1**

**currneighs -= removed\_neighs**

**currchilds -= removed\_childs**

**#all edges covered?**

**if len(currneighs) == 0 and state == UNDEC:**

**state = NOTCOV**

**round\_saved = roun**

**if len(currchilds) == 0:**

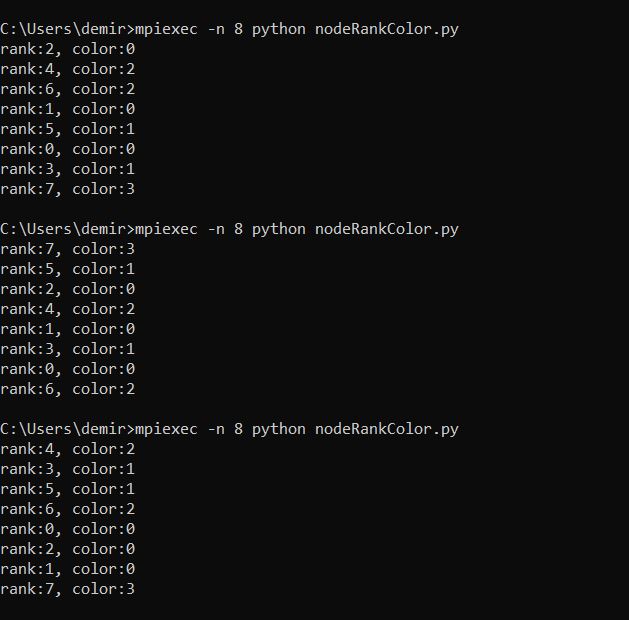
**finished\_all = True**

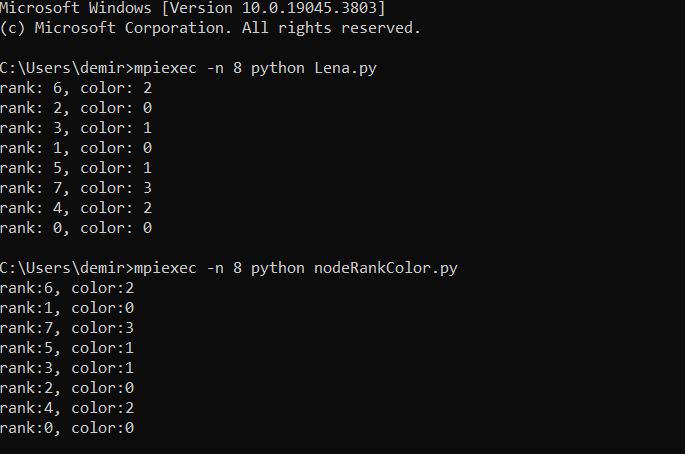
**msg[1] = FINISHED**

**comm.send(msg,dest=parent,tag=FINISHED)**

**print(" Rank: {}, color: {}".format(rank,round\_saved))**

**Output**

****

****

**Pseudocode**

**1: int rank ← MPI\_Comm\_rank, n ← MPI\_Comm\_size**

**2: enum states {UNDEC, INCOV, NOTCOV}**

**3: enum message\_types {ROUND, DATA, ROVER, FINISHED, TERMINATE}**

**4: array of int msg ← [⊥, ⊥, ⊥, ⊥]**

**5: matrix int A[n][n] ← ... // adjacency matrix**

**6: list of list of int children ←[[3, 4, 7], [], [6], [2], [], [1], [],[5]]**

**7: list of int parents ← [0, 5, 3, 0, 0, 7, 2, 0]**

**8: set of int childs, neighs, neighs\_rcvd, rover\_rcvd, finish\_rcvd ← ∅**

**9: set of int removed\_neighs, removed\_childs ← ∅**

**10: list of states states ← [UNDEC, INCOV, NOTCOV]**

**11: int state ← UNDEC**

**12: int round\_num ← 0, max\_count ← 4**

**13: boolean finished\_childs ← false, finished\_all ← false, terminated ← false**

**14:**

**15: for int i ← 0 to n-1 do**

**16: if A[rank][i] = 1 then**

**17: neighs ← neighs ∪ {i}**

**18: end if**

**19: end for**

**20: int parent ← parents[rank]**

**21: set of int currneighs ← neighs, currchilds ← children[rank]**

**22:**

**23: while ¬terminated do**

**24: boolean round\_over ← false, neighs\_over ← false, childs\_over ← false**

**25: boolean neighs\_sent ← false**

**26: rover\_rcvd.clear, neighs\_rcvd.clear**

**27: removed\_neighs.clear, removed\_childs.clear**

**28:**

**29: if rank = 0 then**

**30: msg[0], msg[1], msg[2] ← rank, ROUND, round\_num**

**31:**

**32: for each child in currchilds do**

**33: MPI\_Send msg to child with tag ROUND**

**34: end for**

**35:**

**36: if state = UNDEC then**

**37: msg[1], msg[3] ← DATA, state**

**38: for each node in currneighs do**

**39: MPI\_Send msg to node with tag DATA**

**40: end for**

**41: neighs\_sent ← true**

**42: else**

**43: neighs\_over ← true, neighs\_sent ← true**

**44: end if**

**45:**

**46: while ¬round\_over do**

**47: msg ← MPI\_Recv from MPI\_ANY\_SOURCE with tag MPI\_ANY\_TAG**

**48:**

**49: switch msg[1] do**

**50: case ROUND: // Handle ROUND messages**

**51: case DATA: // Handle DATA messages**

**52: case ROVER: // Handle ROVER messages**

**53: case FINISHED: // Handle FINISHED messages**

**54: case TERMINATE: // Handle TERMINATE messages**

**55: end switch**

**56: end while**

**57:**

**58: round\_num ← round\_num + 1**

**59: currneighs ← currneighs - removed\_neighs**

**60: currchilds ← currchilds - removed\_childs**

**61:**

**62: if currneighs = ∅ and state = UNDEC then**

**63: state ← NOTCOV**

**64:**

**65: if currchilds = ∅ then**

**66: finished\_all ← true**

**67: msg[1] ← FINISHED**

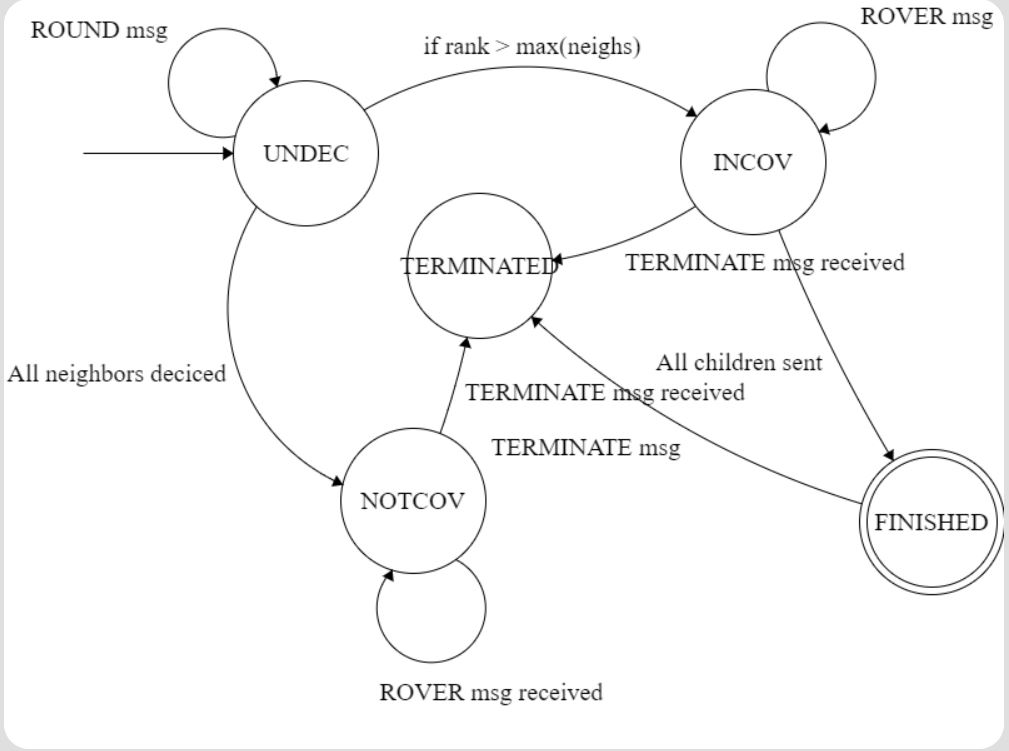
**68: MPI\_Send msg to parent with tag FINISHED**

**69: end if**

**70: end if**

**71: end while**

**72: print "Rank: ", rank, ", state: ", state**

**FSM Diagram for one node:  
**

**FSM table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Current**  **State** | **Event**  **(Message Type)** | **Condition/Action** | **New State** |
| **UNDEC** | **ROUND** | **-Received ROUND message.** | **(No Change)** |
| **UNDEC** | **DATA** | **- Rank greater than max(current neighbors).** | **INCOV** |
| **UNDEC** | **DATA** | **- All neighbors decided.** | **NOTCOV** |
| **INCOV** | **ROVER** | **- Received ROVER message indicating child nodes have finished.** | **(No Change)** |
| **INCOV** | **FINISHED** | **- All child nodes are finished.** | **TERMINATED** |
| **NOTCOV** | **ROVER** | **- Received ROVER message indicating child nodes have finished.** | **(No Change)** |
| **NOTCOV** | **FINISHED** | **- All child nodes are finished.** | **TERMINATED** |
| **INCOV** | **TERMINATE** | **- Received TERMINATE message.** | **TERMINATED** |
| **NOTCOV** | **TERMINATE** | **- Received TERMINATE message.** | **TERMINATED** |
| **UNDEC** | **FINISHED** | **- If in a state where finishing is valid (all neighbors decided).** | **FINISHED** |
| **UNDEC** | **TERMINATE** | **- Received TERMINATE message.** | **TERMINATED** |