

Terna Engineering College
Computer Engineering Department
Program: Sem VIII

Course: Distributed Computing Lab (CSL802)

Faculty: Rohini Patil

Experiment No. 7

A.1 Aim: To Implement Chandi-Misra-Haas distributed deadlock detection algorithm.

PART B
(PART B: TO BE COMPLETED BY STUDENTS)

Roll No. 50	Name: AMEY MAHENDRA THAKUR
Class: BE COMPS B 50	Batch: B3
Date of Experiment: 25-02-2022	Date of Submission: 25-02-2022
Grade:	

B.1 Software Code written by a student:

- ChandyMisraHaas.java

No Deadlock:

```
print("""
P1 P2 P3 P4 P5
```

```
P1 0 1 0 0 0
P2 0 0 1 0 0
P3 0 0 0 1 1
P4 0 0 0 0 0
P5 0 0 0 0 0
""")
```

```
a = [ [0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0],
       [0, 0, 0, 1, 1],
```

```
[0, 0, 0, 0, 0],
[0, 0, 0, 0, 0] ]
```

Deadlock:

```
print("""
P1 P2 P3 P4 P5

P1 0 1 0 0 0
P2 0 0 1 0 0
P3 0 0 0 1 1
P4 1 0 0 0 0
P5 0 0 0 0 0
""")
```

```
a = [ [0, 1, 0, 0, 0],
```

```
[0, 0, 1, 0, 0],
[0, 0, 0, 1, 1],
```

```
[1, 0, 0, 0, 0],
[0, 0, 0, 0, 0] ]
```

```
flag = 0
```

```
def aman(a, i, k):
```

```
    end = 5
```

```
    for x in range(end):
```

```
        if(a[k][x] == 1):
```

```
            if(i == x):
```

```
                print(f' S{k+1} ==> S{x+1}    ({i+1}, {k+1}, {x+1}) ----->
```

```
                DEADLOCK DETECTED')
```

```
    global flag
```

```
    flag = 1
```

```
    break
```

```
    print(f' S{k+1} ==> S{x+1}    ({i+1}, {k+1}, {x+1})')
```

```
    aman(a,i,x)
```

```
print("CHANDY-MISRA-HAAS DISTRIBUTED DEADLOCK DETECTION ALGORITHM")
```

```
print("-----")
```

```
print()
```

```
x = 0
```

```
end = 5
```

```
i = int(input("Enter Initiator Site No. : "))
```

```
j = i - 1
```

```
print()
```

```
for k in range(end):
```

```
    if(a[j][k]==1):
```

```
        print(f' S{j+1} ==> s{k+1}    ({i}, {j+1}, {k+1})')
```

```
        aman(a,j,k)
```

```
if(flag == 0):
```

```
    print("\nNO DEADLOCK DETECTED")
```

```
print("-----")
```

B.2 Input and Output:

- No Deadlock:

```
Command Prompt

C:\Users\ameyt\Desktop>python ChandyMisraHaas.py

      P1  P2  P3  P4  P5
P1  0   1   0   0   0
P2  0   0   1   0   0
P3  0   0   0   1   1
P4  0   0   0   0   0
P5  0   0   0   0   0

CHANDY-MISRA-HAAS DISTRIBUTED DEADLOCK DETECTION ALGORITHM

Enter Initiator Site No. : 1

S1 ==> s2      (1, 1, 2)
S2 ==> S3      (1, 2, 3)
S3 ==> S4      (1, 3, 4)
S3 ==> S5      (1, 3, 5)

NO DEADLOCK DETECTED
```

- Deadlock:

```
Command Prompt

C:\Users\ameyt\Desktop>python ChandyMisraHaas.py

      P1  P2  P3  P4  P5
P1  0   1   0   0   0
P2  0   0   1   0   0
P3  0   0   0   1   1
P4  1   0   0   0   0
P5  0   0   0   0   0

CHANDY-MISRA-HAAS DISTRIBUTED DEADLOCK DETECTION ALGORITHM

Enter Initiator Site No. : 1

S1 ==> s2      (1, 1, 2)
S2 ==> S3      (1, 2, 3)
S3 ==> S4      (1, 3, 4)
S4 ==> S1      (1, 4, 1) -----> DEADLOCK DETECTED
S3 ==> S5      (1, 3, 5)
```

B.3 Observations and learning:

- This is considered an edge-chasing, probe-based algorithm.
- It is also considered one of the best deadlock detection algorithms for distributed systems.
- If a process makes a request for a resource that fails or times out, the process generates a probe message and sends it to each of the processes holding one or more of its requested resources.
- Each probe message contains the following information:
 - the id of the process that is blocked (the one that initiates the probe message);
 - the id of the process is sending this particular version of the probe message;
 - the id of the process that should receive this probe message.
- When a process receives a probe message, it checks to see if it is also waiting for resources.
- If not, it is currently using the needed resource and will eventually finish and release the resource.
- If it is waiting for resources, it passes on the probe message to all processes it knows to be holding resources it has itself requested.
- The process first modifies the probe message, changing the sender and receiver ids.
- If a process receives a probe message that it recognizes as having initiated, it knows there is a cycle in the system and thus, deadlock.
- The advantages of this algorithm include the following:
 - It is easy to implement.
 - Each probe message is of fixed length.
 - There is very little computation.
 - There is very little overhead.
 - There is no need to construct a graph, nor to pass graph information to other sites.
 - This algorithm does not find false (phantom) deadlock.
 - There is no need for special data structures.

B.4 Conclusion:

Successfully implemented Chandy-Misra-Haas distributed deadlock detection algorithm using python.

B.5 Question of Curiosity:

Q1: Consider the following statements: A deadlock detection algorithm must satisfy the following two conditions:

Condition 1: Progress (No false deadlocks): The algorithm should not report deadlocks that do not exist.

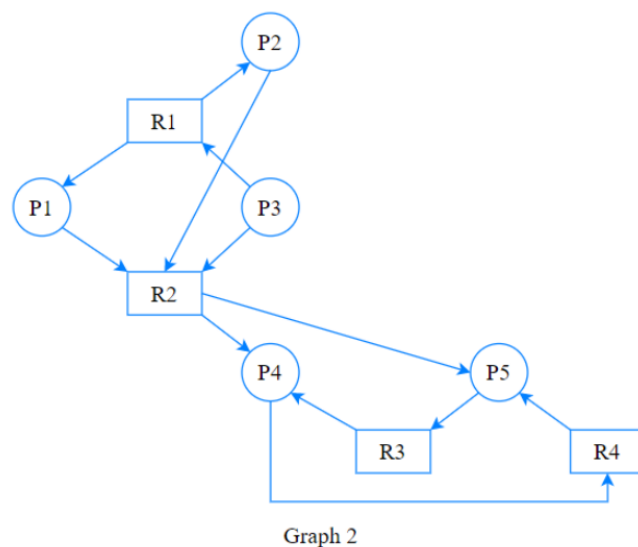
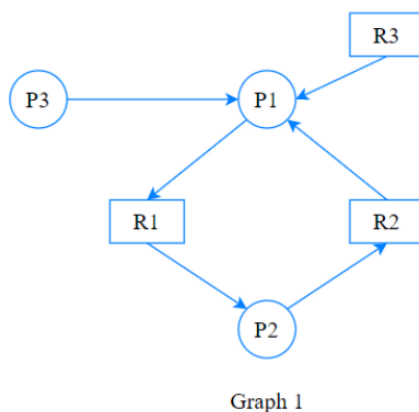
Condition 2: Safety (No undetected deadlocks): The algorithm must detect all existing deadlocks infinite time.

A. Both conditions are true

- B. Both conditions are false
- C. Only condition 1 is true
- D. Only condition 1 is true

ANS: A. Both conditions are true

Q2: Consider the resource allocation graphs G1 and G2. What can be said about the deadlock conditions?



A. Both are deadlocked

- B. G1 deadlocked, G2 not
- C. G1 not, G2 deadlocked
- D. None is deadlocked

ANS: A. Both are deadlocked

Q3: Which of the following problem we might face if we invoke the deadlock detection algorithm at the arbitrary interval?

- A. There may be many cycles in the graph and it will not be possible to roll back the deadlocked processes anymore.
- B. There may be many cycles in the graph and we would not be able to tell which of the many deadlocked processes caused the deadlock.
- C. There may be no cycles in the graph and thus it will not be possible to know if any deadlock has happened.
- D. All of the above.

ANS: B. There may be many cycles in the graph and we would not be able to tell which of the many deadlocked processes caused the deadlock

Q4: Consider the following statements related to distributed deadlock detection algorithms:

Statement 1: In path-pushing algorithms, distributed deadlocks are detected by maintaining an explicit global WFG. The basic idea is to build a global WFG for each site of the distributed system.

Statement 2: In an edge-chasing algorithm, the presence of a cycle in a distributed graph structure is be verified by propagating special messages called probes, along the edges of the graph. These probe messages are different from the request and reply messages.

- A. Both statements are true
- B. Both statements are false
- C. Only statement 1 is true
- D. Only statement 2 is true

ANS: A. Both statements are true