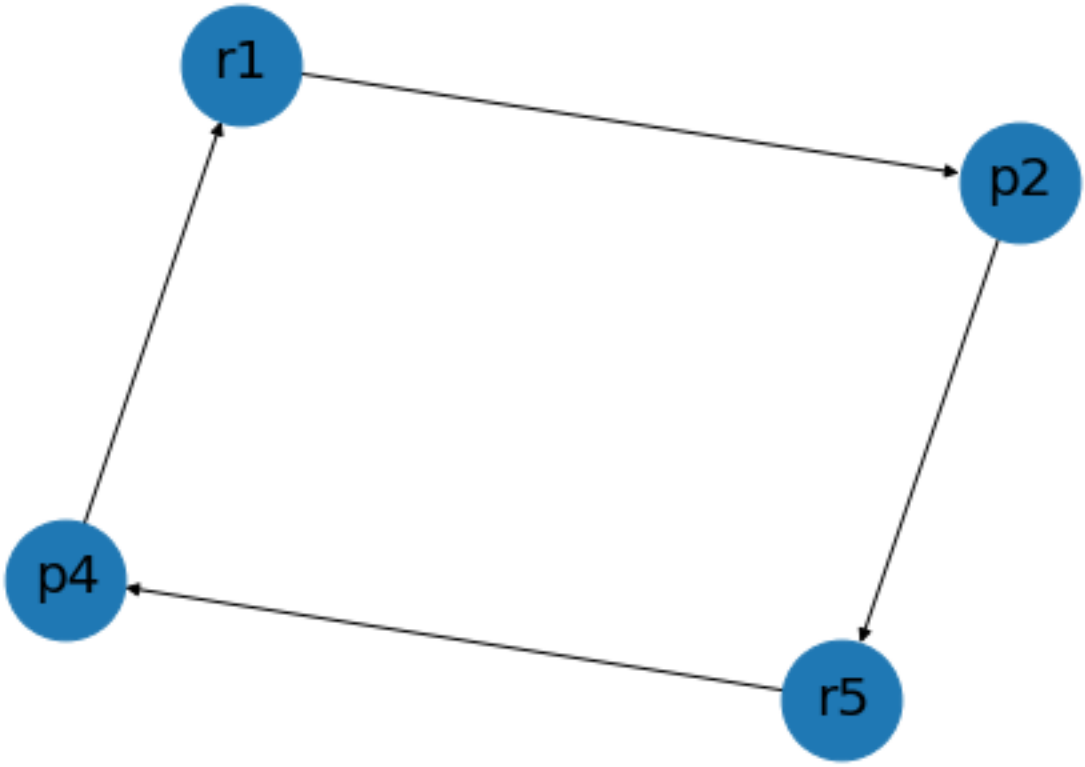


Deadlock Detection using python

Mohaned Ashraf Abdo

Section 1

Third Year

Deadlock Detection

In this case for Deadlock detection we can run an algorithm to check for cycle in the Resource Allocation Graph. Presence of cycle in the graph is enough condition for deadlock.

Let’s look at the code

```python
from collections import defaultdict
from enum import Enum
import networkx as nx
import matplotlib.pyplot as plt

class Graph:
    def __init__(self):
        self.graph = defaultdict(list) #value for every key in dictonary graph is a list
        self.cycleCount = 0 #count number of cycles in graph
    def addEdge(self,source,target):
        self.graph[source].append(target) #add an directed edge from source to target

    #Thoose are the 3 node states that we got
    class NodeState(Enum):
        NOT_VISITED = 1
        IN_CURRENT_PATH =2
        PROCESSED_BEFORE =3

    #The graph can contain more than connected component
    def getAllCyclesInGraph(self):
        visited = {}
        #for every connected component in graph get all cycles
        for node in self.graph:
            if not visited.get(node) :
                self.getAllCyclesInConnectedComponent(node,visited)
```

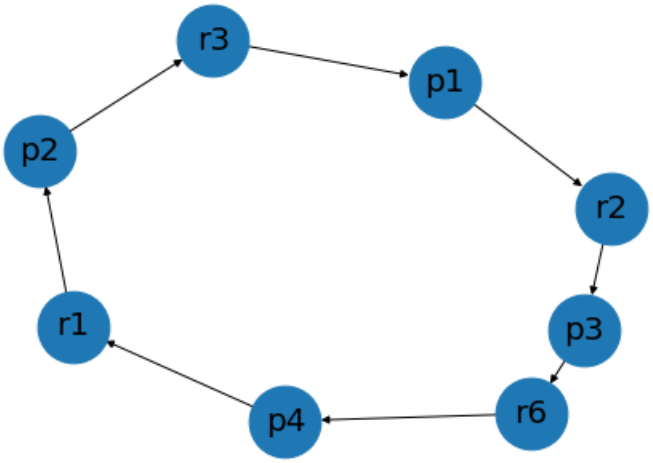
```
def getAllCyclesInConnectedComponent(self,node,visited,path=[]):
        # if the node is visited and in the current path so we just found our cycle
        if visited.get(node) and visited[node] == self.NodeState.IN_CURRENT_PATH:
            #just for outputting the cycle
            foundStartOfCycle =False
            self.cycleCount = self.cycleCount +1
            print("Deadlock number {} was found\n".format(self.cycleCount))
            G=nx.DiGraph()
            G.clear()

            startPostionOfCycle = 0
            for i in range(0,len(path)):
                #search for the start of the cycle
                if(path[i] == node):
                    foundStartOfCycle= True
                    startPostionOfCycle = i
                if foundStartOfCycle and path[i]:
                    G.add_node(path[i])
                    if(i+1==len(path)):
                        G.add_edge(path[i],path[startPostionOfCycle])
                    else:
                        G.add_edge(path[i],path[i+1])

            nx.draw(G,with_labels = True,font_size=20,node_size=2000)
            plt.savefig("Deadlock{}".format(self.cycleCount))

            plt.show()
            return

        #if no cycle is found yes
        visited[node]= self.NodeState.IN_CURRENT_PATH #mark node as being processing now
        path.append(node) #add node to current path

        #go to every possible child from node (depth first search)
        for child in self.graph[node]:
            if not visited.get(child) or  visited[child]==self.NodeState.IN_CURRENT_PATH:
                self.getAllCyclesInConnectedComponent(child,visited,path)

        path.pop() #remove node from current path
        visited[node]= self.NodeState.PROCESSED_BEFORE #mark node as processed

```

```python
class DeadlockChecker:

    def __init__(self):
        self.graph = Graph()

    def processHaveResource(self,processID,resourceID):
        #this for handling user input only id with no string prefix
        if type(processID)== int:
            processID = 'p'+str(processID)
        if type(resourceID)==int:
            resourceID = 'r' + str(resourceID)
        self.graph.addEdge(resourceID,processID)

    def processWaitResource(self,processID,resourceID):
        #this for handling user input only id with no string prefix

        if type(processID)== int:
            processID = 'p'+str(processID)
        if type(resourceID)==int:
            resourceID = 'r' + str(resourceID)
        self.graph.addEdge(processID,resourceID)

    def checkForDeadlock(self):
        self.graph.getAllCyclesInGraph()
```

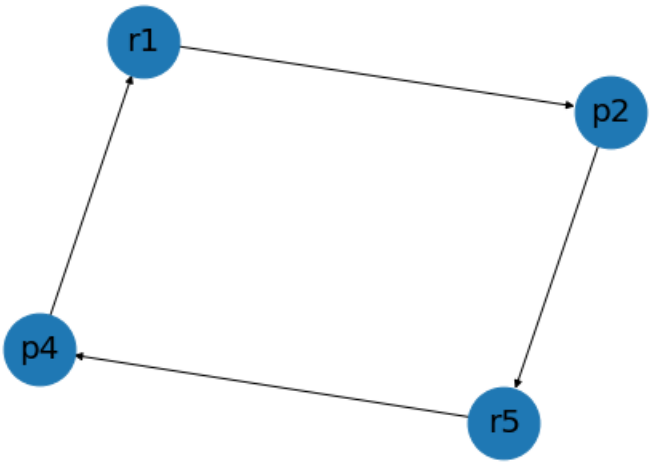
```python
def main():
    deadlockChecker = DeadlockChecker()

    #comment to disable manual input
    #example from my OS lecture
    deadlockChecker.processHaveResource(1,3)
    deadlockChecker.processHaveResource(1,4)
    deadlockChecker.processHaveResource(2,1)
    deadlockChecker.processHaveResource(3,2)
    deadlockChecker.processHaveResource(4,5)
    deadlockChecker.processHaveResource(4,6)
    deadlockChecker.processHaveResource(5,8)
    deadlockChecker.processHaveResource(6,7)

    deadlockChecker.processWaitResource(1,2)
    deadlockChecker.processWaitResource(2,3)
    deadlockChecker.processWaitResource(2,5)
    deadlockChecker.processWaitResource(3,6)
    deadlockChecker.processWaitResource(3,1)
    deadlockChecker.processWaitResource(4,1)
    deadlockChecker.processWaitResource(4,7)
    deadlockChecker.processWaitResource(5,5)
    deadlockChecker.processWaitResource(5,3)
    deadlockChecker.processWaitResource(6,8)
    deadlockChecker.processWaitResource(6,6)

    deadlockChecker.checkForDeadlock()


     #uncomment to take input from user
#    while(True):
#        print("please insert the process ID then resource ID then 1 for have and 2 for wait, all on oneline")
#        print("please insert 0 0 0 to terminate program and see results")
#
#        pID,rID,edgeType = map(int, input().split())
#        if edgeType == 1:
#            deadlockChecker.processHaveResource(pID,rID)
#        elif edgeType == 2:
#            deadlockChecker.processWaitResource(pID,rID)
#        elif edgeType ==0 and pID ==0  and rID ==0:
#            deadlockChecker.checkForDeadlock()
#            break
#        else:
#            print("please insert valid type or ")
#            print("please insert 0 0 0 to terminate program and see results")
#            continue


if __name__ == '__main__':
    main()
```

And here is the output

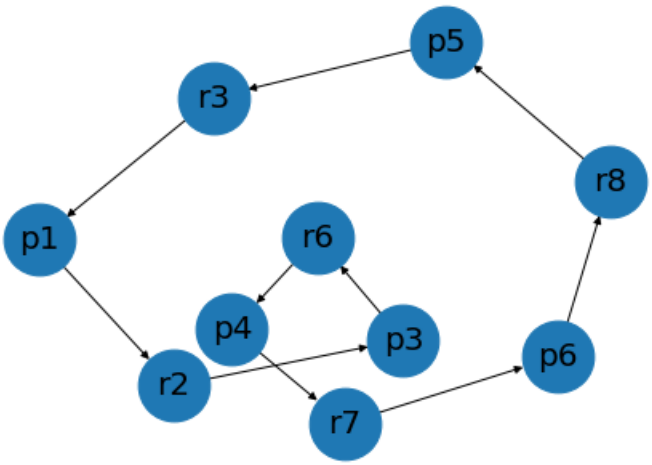
```
In [6]: runfile('D:/OS/DeadlockDetection.py', wdir='D:/OS')
Deadlock number 1 was found
```




```
Deadlock number 2 was found
```



```
Deadlock number 3 was found
```




```
Deadlock number 4 was found
```