PA4

Jordan Taranto

Code

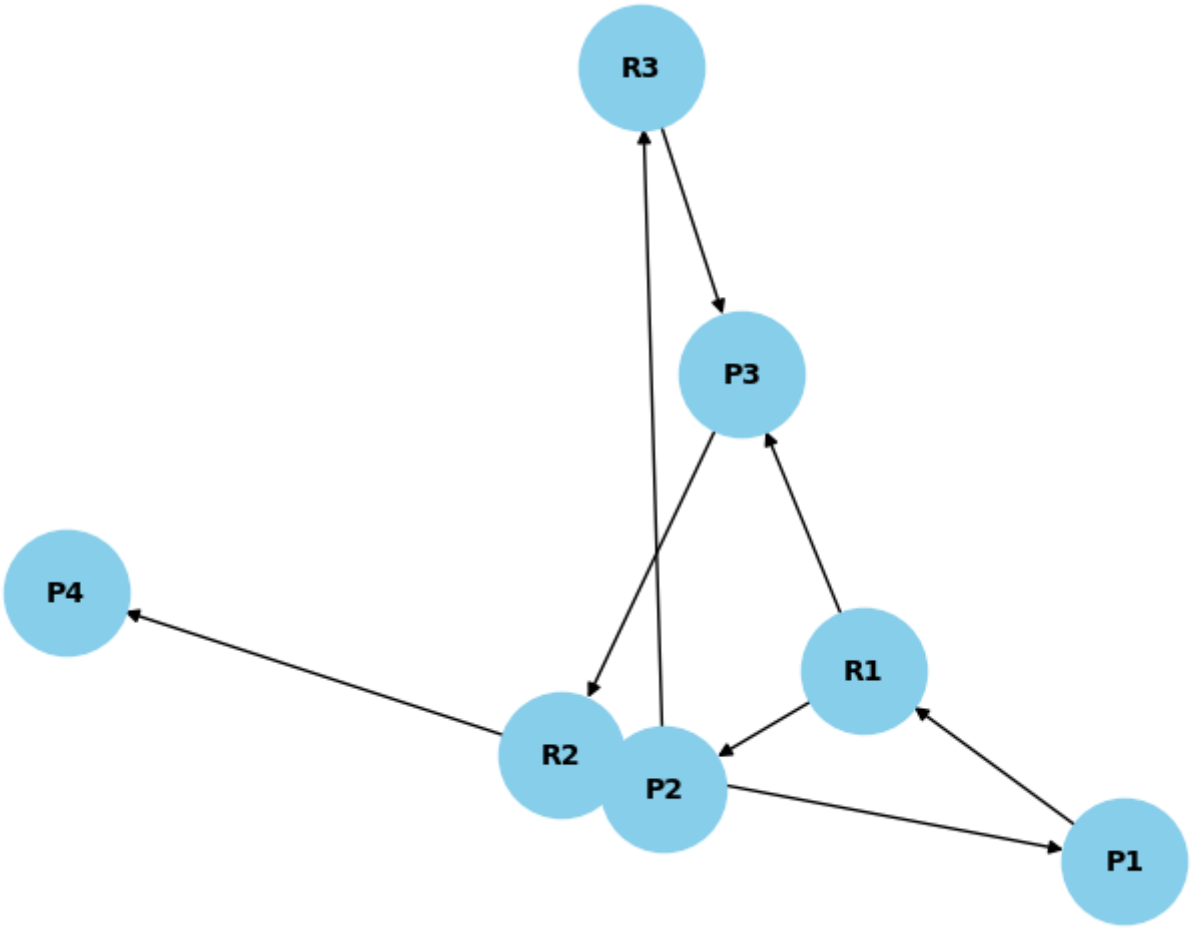
```
# author: Jordan Taranto
import matplotlib.pyplot as plt
import networkx as nx

# RAG allocation graph class implementation
class RAG:
def __init__(self, processes_list, resources_list):

# init an empty directed graph
self.graph = nx.DiGraph()

# store list of process and resources
self.processes_list = processes_list
self.resources_list = resources_list

def add_edge(self, process, resource):
# add edge from process to resource
self.graph.add_edge(process, resource)

def visualize(self):

# generate layout positions for the processes

layout = nx.spring_layout(self.graph)



# draw the graph

nx.draw(self.graph, layout, with_labels=True, node_color='skyblue',
node_size=2000, font_size=10, font_weight='bold')



def detect_deadlock(self):

# find cycles in the graph
```

```
cycles = list(nx.simple_cycles(self.graph))


# check if cycles are found

if cycles:

print('Deadlock detected')

print('Processes involved in the deadlock:', cycles)

else:

print('No deadlocks detected')


plt.title('Resource Allocation Graph')

plt.show()


processes = ['P1', 'P2', 'P3', 'P4']

resources = ['R1', 'R2', 'R3', 'R4']


rag = RAG(processes, resources)


# Graph 1 - no deadlock

rag.add_edge('P1', 'R1')

rag.add_edge('R2', 'P1')

rag.add_edge('R2', 'P2')

rag.add_edge('R1', 'P2')

rag.add_edge('P2', 'R3')
```

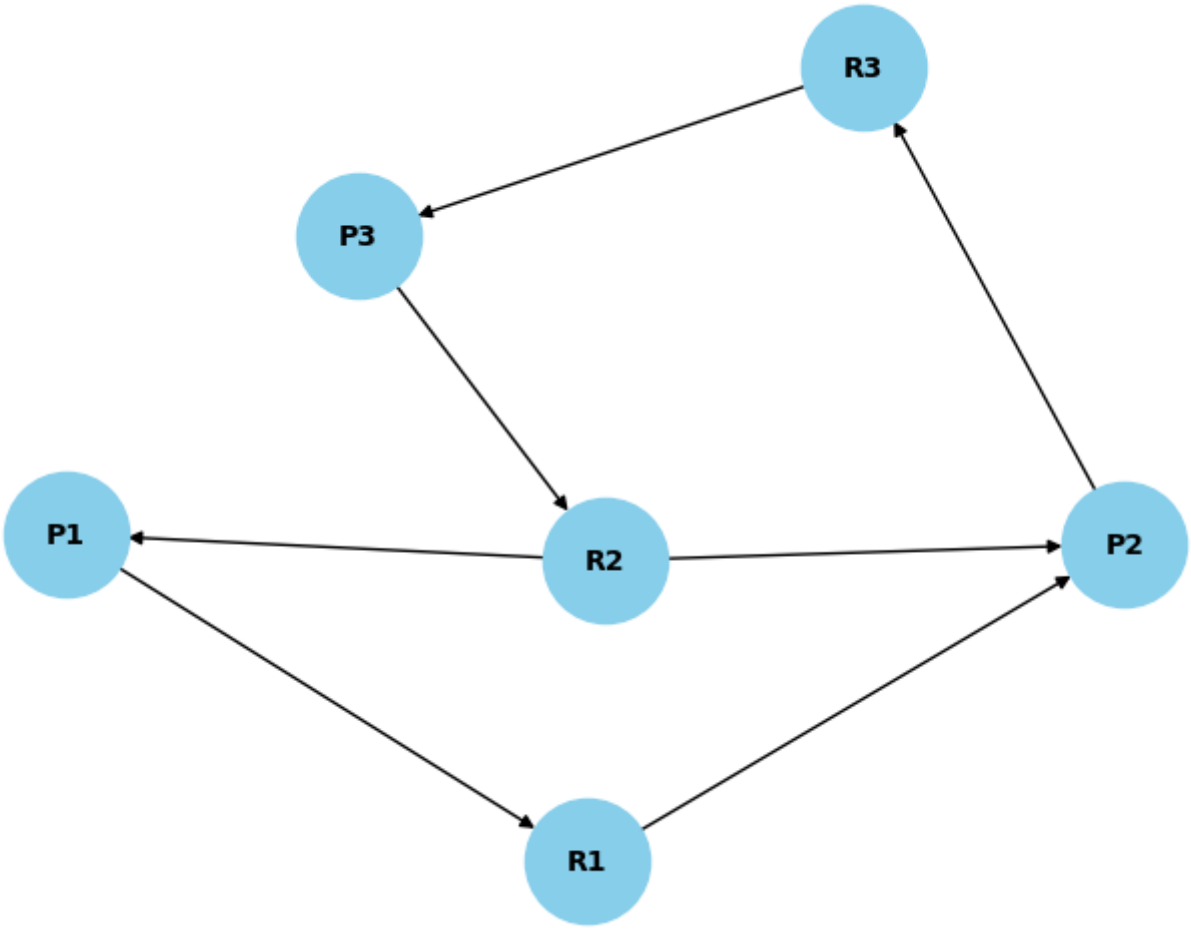
```
rag.add_edge('R3', 'P3')


rag.visualize()
rag.detect_deadlock()


# Graph 2 - deadlock
rag.add_edge('P1', 'R1')
rag.add_edge('R2', 'P1')
rag.add_edge('R2', 'P2')
rag.add_edge('R1', 'P2')
rag.add_edge('P2', 'R3')
rag.add_edge('R3', 'P3')
rag.add_edge('P3', 'R2')


rag.visualize()
rag.detect_deadlock()


# Graph 3 - no deadlock
rag.add_edge('P1', 'R1')
rag.add_edge('R1', 'P2')
rag.add_edge('R1', 'P3')
rag.add_edge('P3', 'R2')
rag.add_edge('R2', 'P1')
```

```
rag.add_edge('R2', 'P4')



rag.visualize()

rag.detect_deadlock()
```

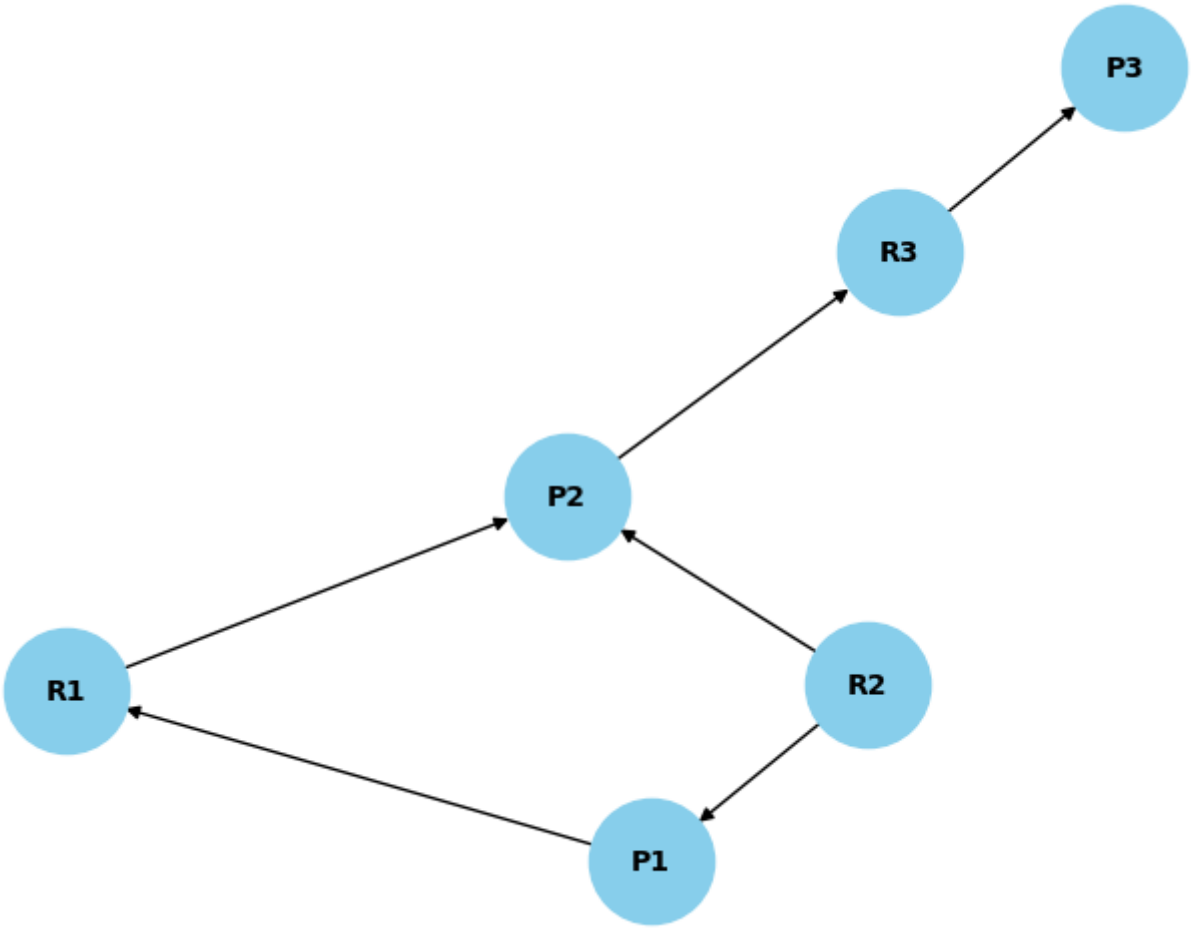
Graph 1 - no deadlock



Graph 2 - deadlock



Graph 3 - no deadlock


P3
R3
P2
R1
R2
P1