Assignment - 6

Aim : Implement an algorithm to detect the deadlock in a given input resource allocation graph.

Theory :

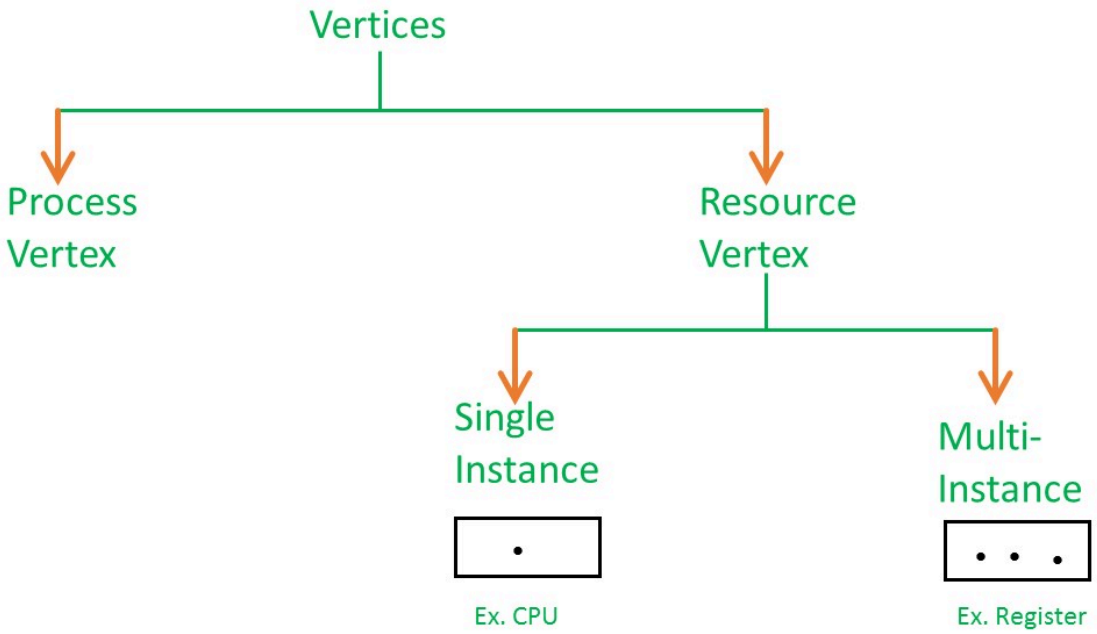
Resource Allocation Graph :

A resource allocation graphs shows which resource is held by which process and which process is waiting for a resource of a specific kind. It is amazing and straight – forward tool to outline how interacting processes can deadlock. Therefore, resource allocation graph describe what the condition of the system as far as process and resources are concern like what number of resources are allocated and what is the request of each process. Everything can be represented in terms of graph. One of the benefit of having a graph is, sometimes it is conveivable to see a deadlock straight forward by utilizing RAG and however you probably won't realize that by taking a glance at the table. Yet tables are better if the system contains bunches of process and resource and graph is better if the system contains less number of process and resource.

1. Process Vertex: Every process will be represented as a process vertex. Generally, the process will be represented with a circle.

2. Resource Vertex: Every resource will be represented as a resource vertex. It is also two types:

Single instance type resource: It represents as a box, inside the box, there will be one dot.So the number of dots indicate how many instances are present of each resource type.
Multi-resource instance type resource: It also represents as a box, inside the box, there will be many dots present.



Program & Output :

```
def is_safe_state(available, rem_need, finished):
    work = available.copy()
    finish = finished.copy()

    while True:
        found = False
        for i in range(no_of_processes):
            if not finish[i]:
                if all(rem_need[i][j] <= work[j] for j in
range(no_of_resources)):
                    for j in range(no_of_resources):
                        work[j] += allocation[i][j]
                    finish[i] = True
                    found = True
        if not found:
            break

    return all(finish)
```

```python
if __name__ == '__main__':
    no_of_processes = 5
    no_of_resources = 3
    total = [10, 5, 7]
    allocation = [
        [0, 1, 0],
        [2, 0, 0],
        [3, 0, 2],
        [2, 1, 1],
        [0, 0, 2]
    ]
    max_need = [
        [7, 5, 3],
        [3, 2, 2],
        [9, 0, 2],
        [2, 2, 2],
        [4, 3, 3]
    ]
    available = [3, 3, 2]
    rem_need = [[0] * no_of_resources for _ in range(no_of_processes)]

    for i in range(no_of_processes):
        for j in range(no_of_resources):
            rem_need[i][j] = max_need[i][j] - allocation[i][j]

    finished = [False] * no_of_processes

    if is_safe_state(available, rem_need, finished):
        print("System is in safe state (No Deadlock detected)\n")
    else:
        print("System is in an unsafe state (Deadlock detected)")
```

```
→  OS git:(master) ✗ python3 rag.py
System is in safe state (No Deadlock detected)
```

Conclusion : Here in this assignment, we studied the resource allocation graph, vertices, types of vertices, edges, and implemented in python using 2d arrays.