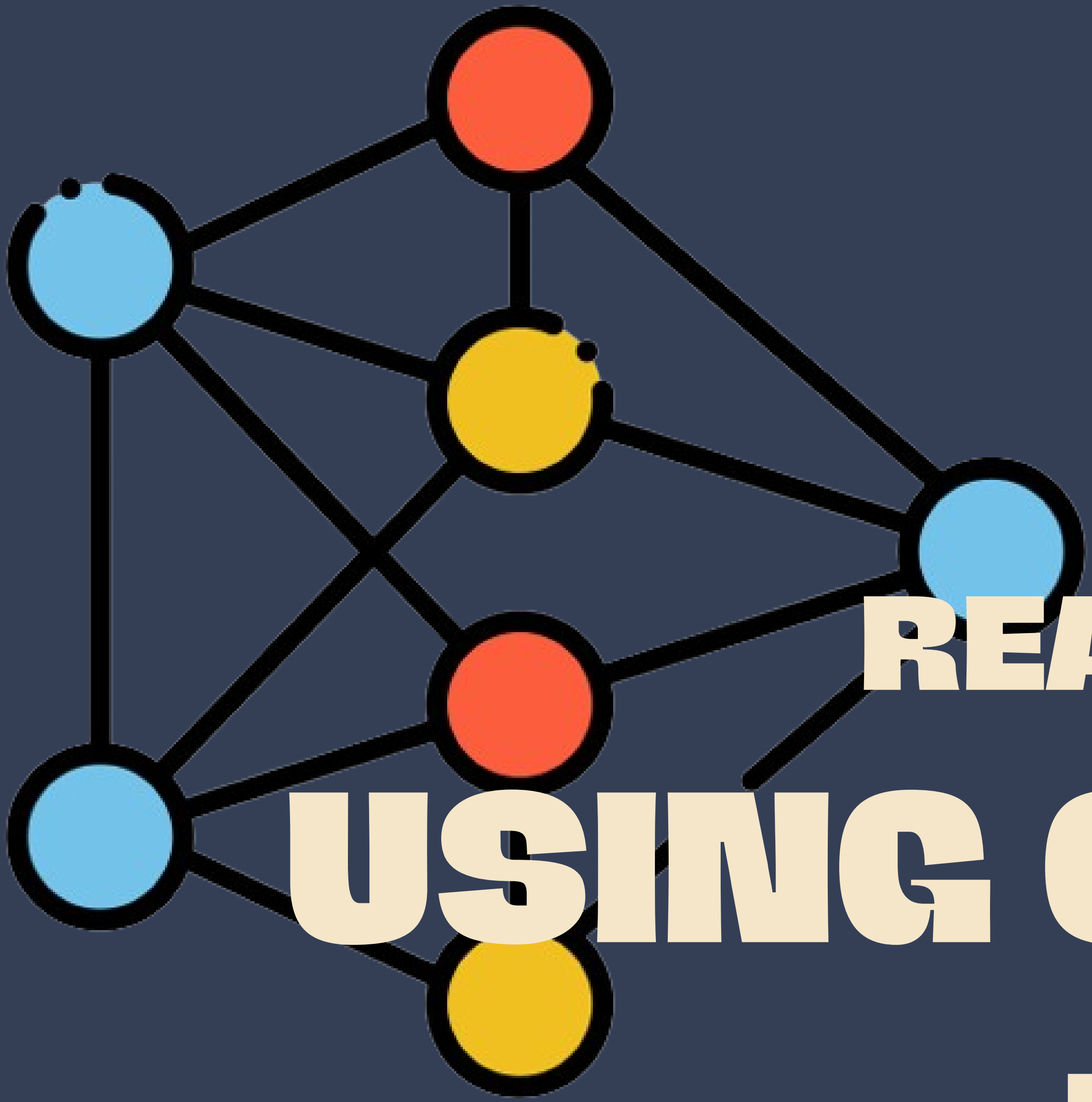
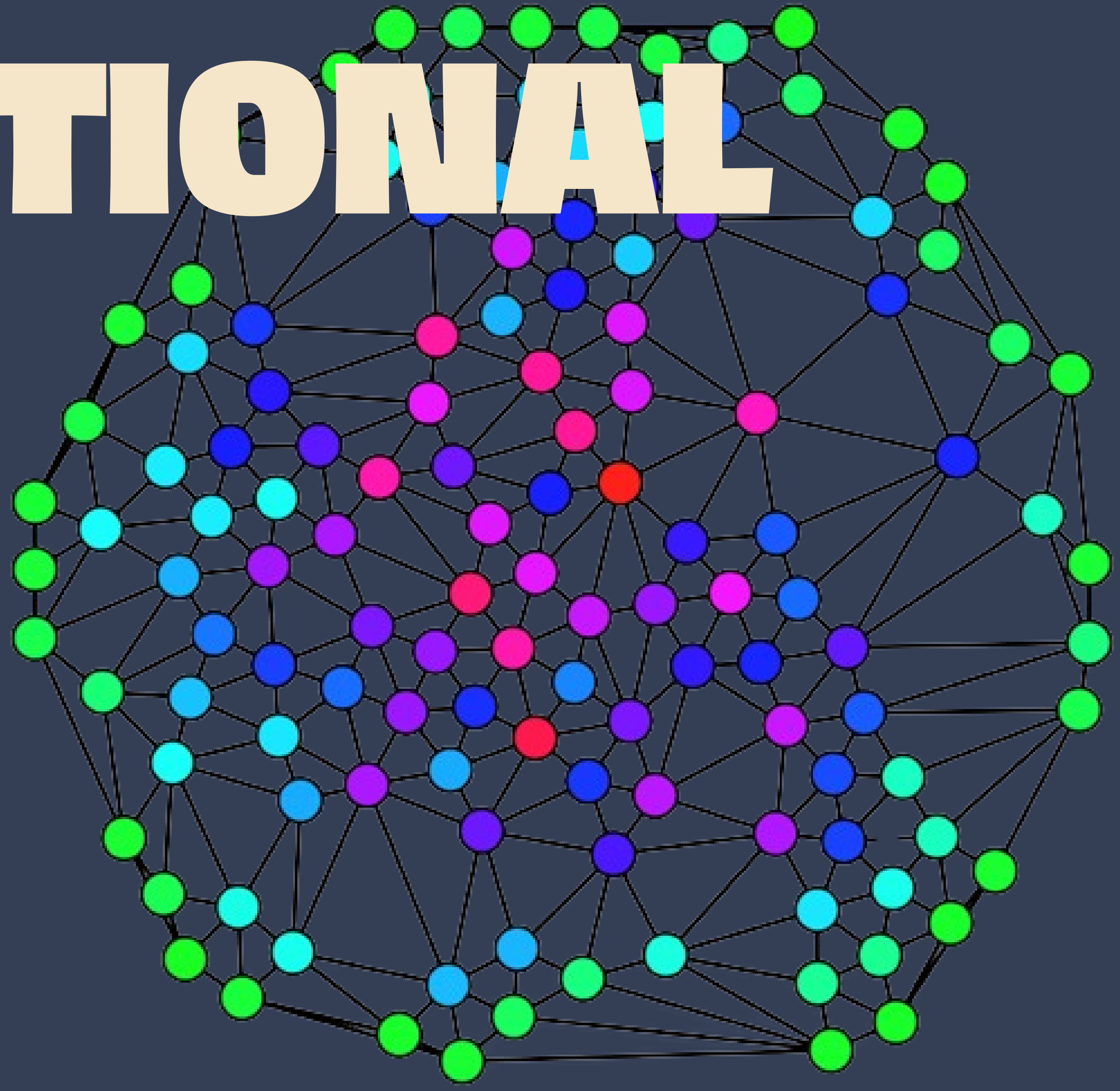


REAL WORLD PRBLEM USING COMPUTATIONAL THINKING

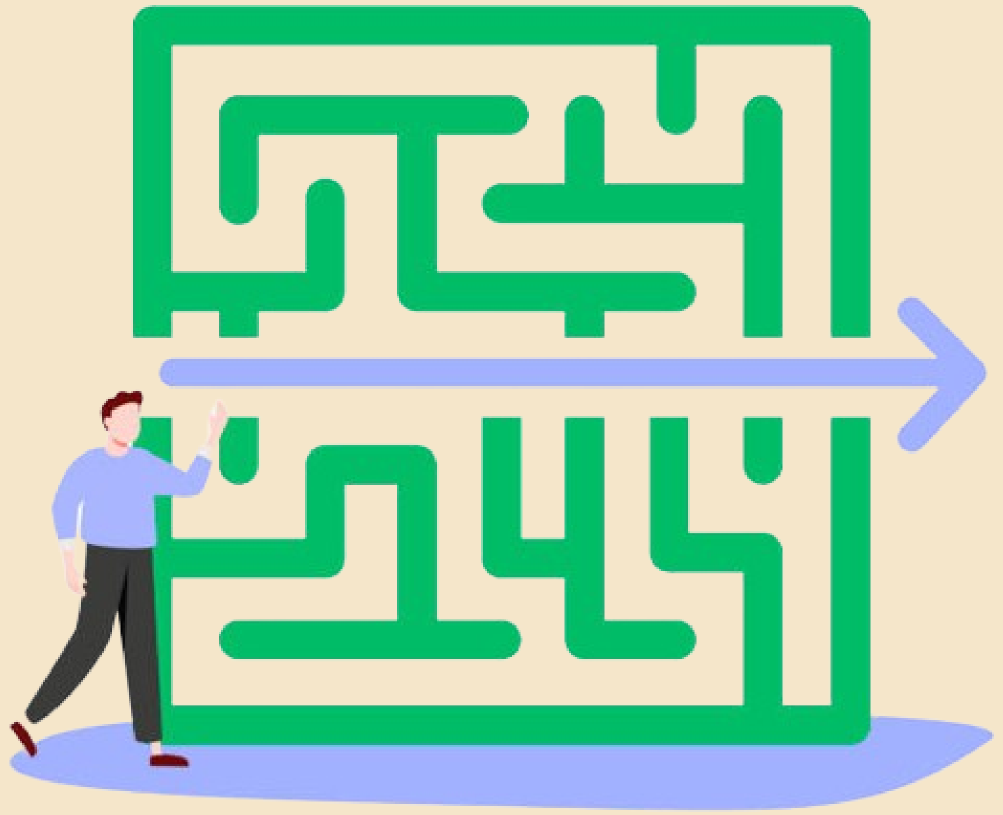


BY TEAM 7

OVERVIEW

INTRODUCTION

When it comes to remembering and navigating the TIP campus, freshman students find it overwhelming to go from one building to another for their next class. Also, they find it difficult to find out what is the shortest route to go from their source to their destination. This leads to problems such as emotional distress and can affect the overall performance and experience of the student such as missing a part of the lecture and attendance.



PROBLEM

Freshmen or Visitors in TIP QC campus encounter difficulties in finding their designated building. Which can end up wasting a lot of their time.

OBJECTIVES

OBJECTIVE 01

To create a program that can navigate students to their designated building for their next class

OBJECTIVE 02

To create an algorithm that can decrease the distance that students must travel in order to reach their assigned building by finding the shortest path from source to destination, particularly for new students or guests who are unfamiliar with a school's layout.

TEAM 7

PROBLEM IDENTIFICATION

How can freshmen or visitors familiarize the campus?

DECOMPOSITION

- Giving them the campus map layout.
- Inviting them into orientation programs.

PATTERN RECOG.

- Freshmen or Newcomers encounter difficulties to find their designated buildings.

ABSTRACTION

Relevant: Buildings
Irrelevant: Freshmen and Visitors

PROBLEM IDENTIFICATION

How can they move from one building to another with minimized time travel?

DECOMPOSITION

- Identifying the location of the buildings.
- Creating a scratch tree graph for the buildings.

PATTERN RECOG.

- When they want to be on time to their schedule.

ABSTRACTION

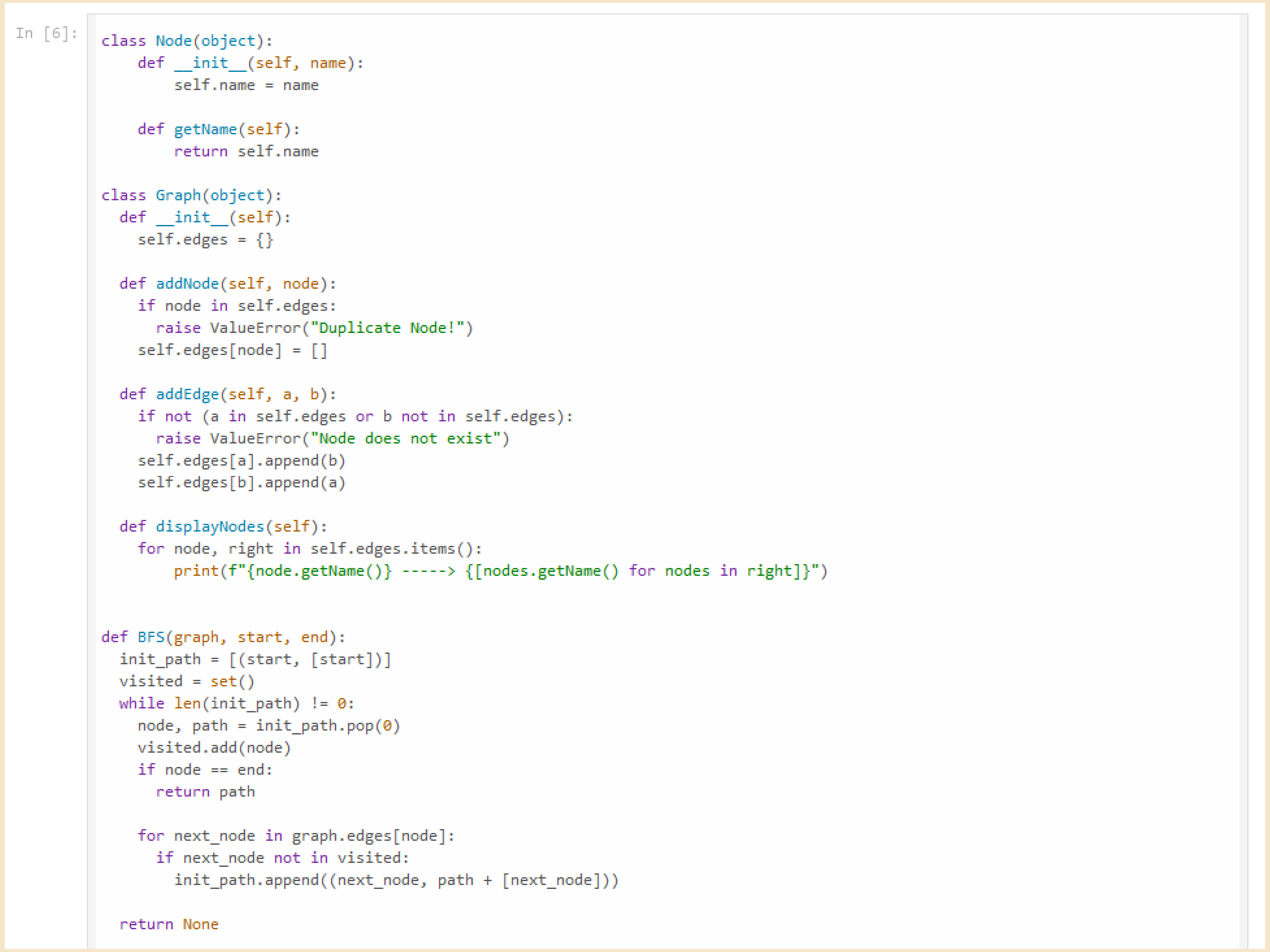
Relevant: Connection of the graph.
Irrelevant: Building size

T

TEAM 7

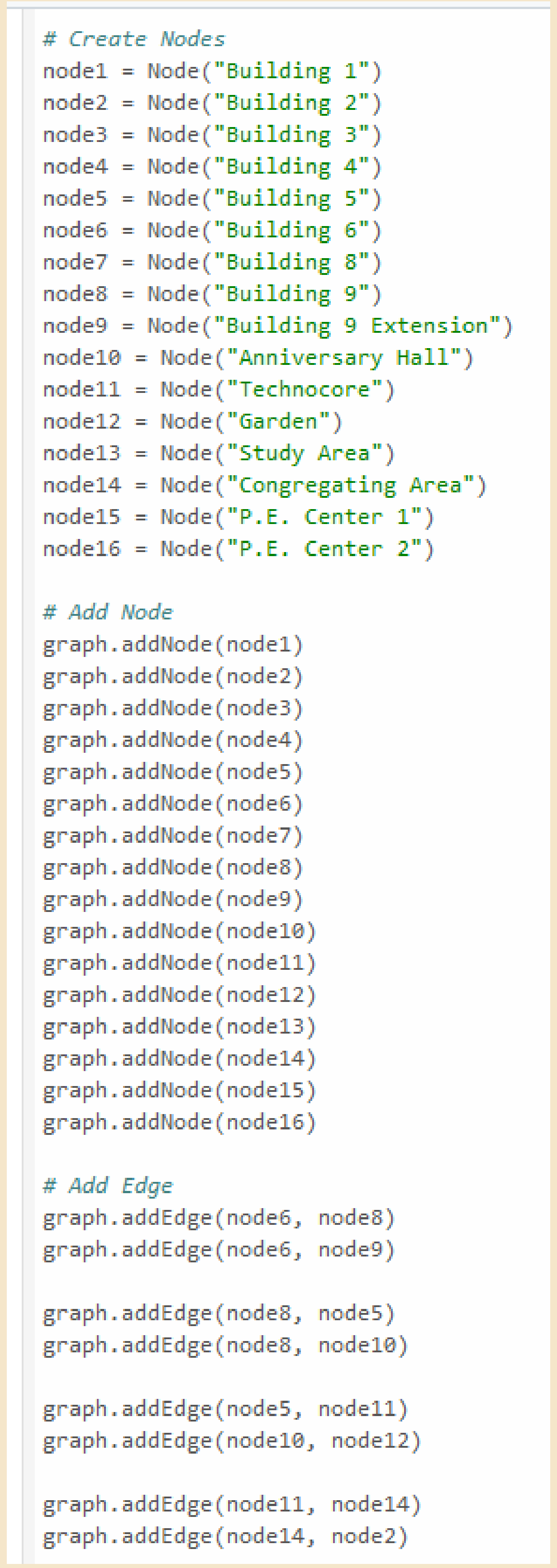
SOLUTION

The solution to this problem is to create a Python code that implements a graph and BFS algorithm to find the shortest path of each building in the campus.

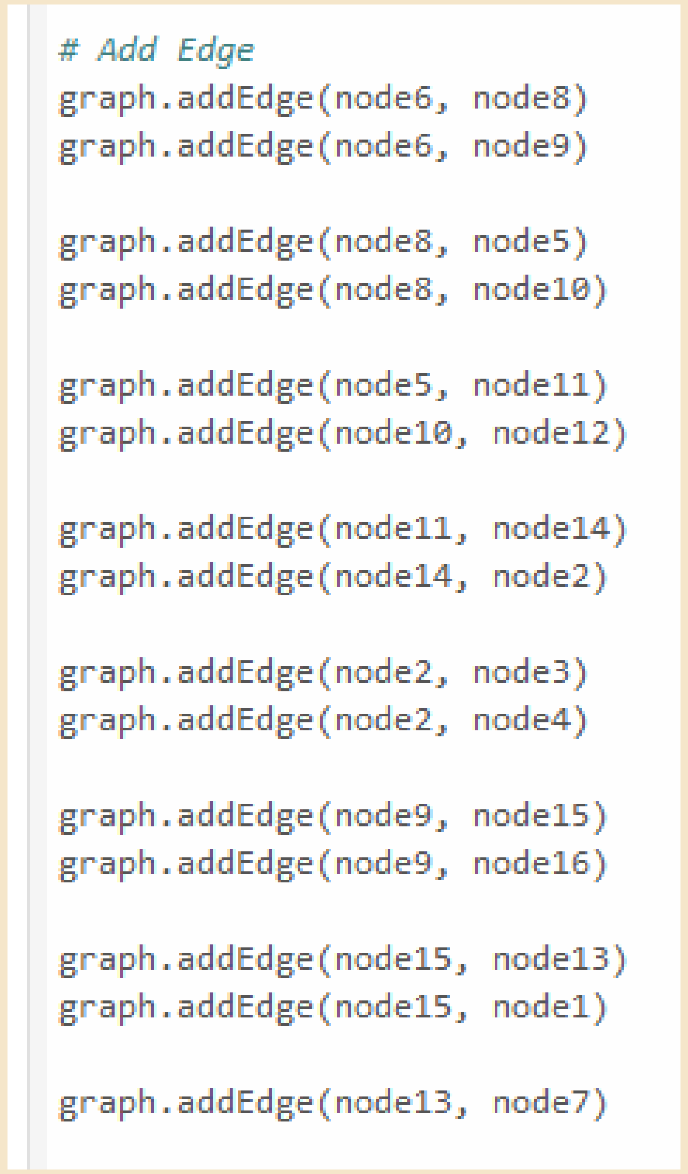


```
In [6]:
```

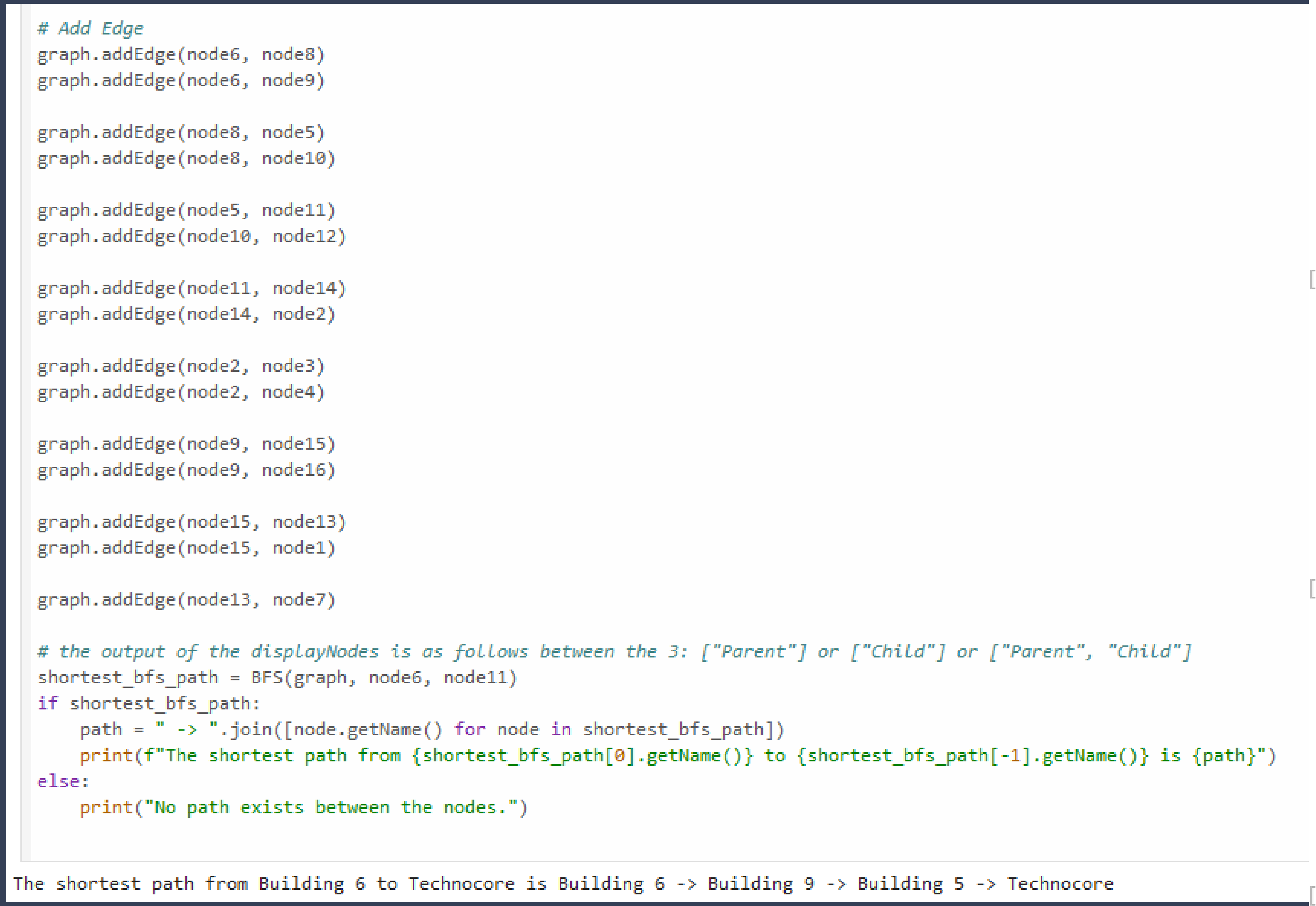
```python
class Node(object):
    def __init__(self, name):
        self.name = name

    def getName(self):
        return self.name

class Graph(object):
  def __init__(self):
    self.edges = {}

  def addNode(self, node):
    if node in self.edges:
      raise ValueError("Duplicate Node!")
    self.edges[node] = []

  def addEdge(self, a, b):
    if not (a in self.edges or b not in self.edges):
      raise ValueError("Node does not exist")
    self.edges[a].append(b)
    self.edges[b].append(a)

  def displayNodes(self):
    for node, right in self.edges.items():
        print(f"{node.getName()} -----> {[nodes.getName() for nodes in right]}")


def BFS(graph, start, end):
  init_path = [(start, [start])]
  visited = set()
  while len(init_path) != 0:
    node, path = init_path.pop(0)
    visited.add(node)
    if node == end:
      return path

    for next_node in graph.edges[node]:
      if next_node not in visited:
        init_path.append((next_node, path + [next_node]))

  return None
```



```python
# Create Nodes
node1 = Node("Building 1")
node2 = Node("Building 2")
node3 = Node("Building 3")
node4 = Node("Building 4")
node5 = Node("Building 5")
node6 = Node("Building 6")
node7 = Node("Building 8")
node8 = Node("Building 9")
node9 = Node("Building 9 Extension")
node10 = Node("Anniversary Hall")
node11 = Node("Technocore")
node12 = Node("Garden")
node13 = Node("Study Area")
node14 = Node("Congregating Area")
node15 = Node("P.E. Center 1")
node16 = Node("P.E. Center 2")

# Add Node
graph.addNode(node1)
graph.addNode(node2)
graph.addNode(node3)
graph.addNode(node4)
graph.addNode(node5)
graph.addNode(node6)
graph.addNode(node7)
graph.addNode(node8)
graph.addNode(node9)
graph.addNode(node10)
graph.addNode(node11)
graph.addNode(node12)
graph.addNode(node13)
graph.addNode(node14)
graph.addNode(node15)
graph.addNode(node16)

# Add Edge
graph.addEdge(node6, node8)
graph.addEdge(node6, node9)

graph.addEdge(node8, node5)
graph.addEdge(node8, node10)

graph.addEdge(node5, node11)
graph.addEdge(node10, node12)

graph.addEdge(node11, node14)
graph.addEdge(node14, node2)
```

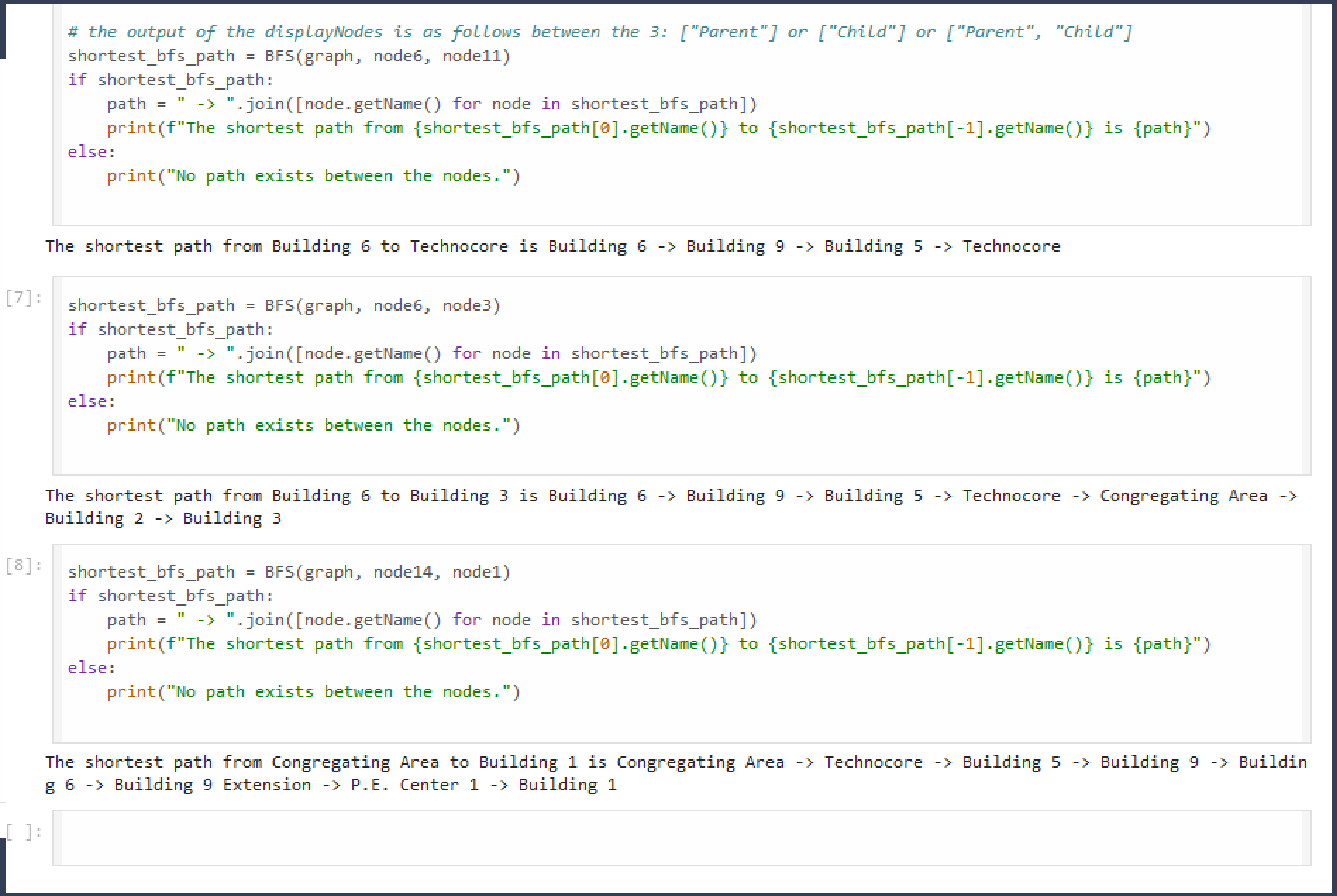


```python
# Add Edge
graph.addEdge(node6, node8)
graph.addEdge(node6, node9)

graph.addEdge(node8, node5)
graph.addEdge(node8, node10)

graph.addEdge(node5, node11)
graph.addEdge(node10, node12)

graph.addEdge(node11, node14)
graph.addEdge(node14, node2)

graph.addEdge(node2, node3)
graph.addEdge(node2, node4)

graph.addEdge(node9, node15)
graph.addEdge(node9, node16)

graph.addEdge(node15, node13)
graph.addEdge(node15, node1)

graph.addEdge(node13, node7)
```



```python
# Add Edge
graph.addEdge(node6, node8)
graph.addEdge(node6, node9)

graph.addEdge(node8, node5)
graph.addEdge(node8, node10)

graph.addEdge(node5, node11)
graph.addEdge(node10, node12)

graph.addEdge(node11, node14)
graph.addEdge(node14, node2)

graph.addEdge(node2, node3)
graph.addEdge(node2, node4)

graph.addEdge(node9, node15)
graph.addEdge(node9, node16)

graph.addEdge(node15, node13)
graph.addEdge(node15, node1)

graph.addEdge(node13, node7)

# the output of the displayNodes is as follows between the 3: ["Parent"] or ["Child"] or ["Parent", "Child"]
shortest_bfs_path = BFS(graph, node6, node11)
if shortest_bfs_path:
    path = " -> ".join([node.getName() for node in shortest_bfs_path])
    print(f"The shortest path from {shortest_bfs_path[0].getName()} to {shortest_bfs_path[-1].getName()} is {path}")
else:
    print("No path exists between the nodes.")
```

```
The shortest path from Building 6 to Technocore is Building 6 -> Building 9 -> Building 5 -> Technocore
```



```python
# the output of the displayNodes is as follows between the 3: ["Parent"] or ["Child"] or ["Parent", "Child"]
shortest_bfs_path = BFS(graph, node6, node11)
if shortest_bfs_path:
    path = " -> ".join([node.getName() for node in shortest_bfs_path])
    print(f"The shortest path from {shortest_bfs_path[0].getName()} to {shortest_bfs_path[-1].getName()} is {path}")
else:
    print("No path exists between the nodes.")
```

```
The shortest path from Building 6 to Technocore is Building 6 -> Building 9 -> Building 5 -> Technocore
```

[7]:

```python
shortest_bfs_path = BFS(graph, node6, node3)
if shortest_bfs_path:
    path = " -> ".join([node.getName() for node in shortest_bfs_path])
    print(f"The shortest path from {shortest_bfs_path[0].getName()} to {shortest_bfs_path[-1].getName()} is {path}")
else:
    print("No path exists between the nodes.")
```

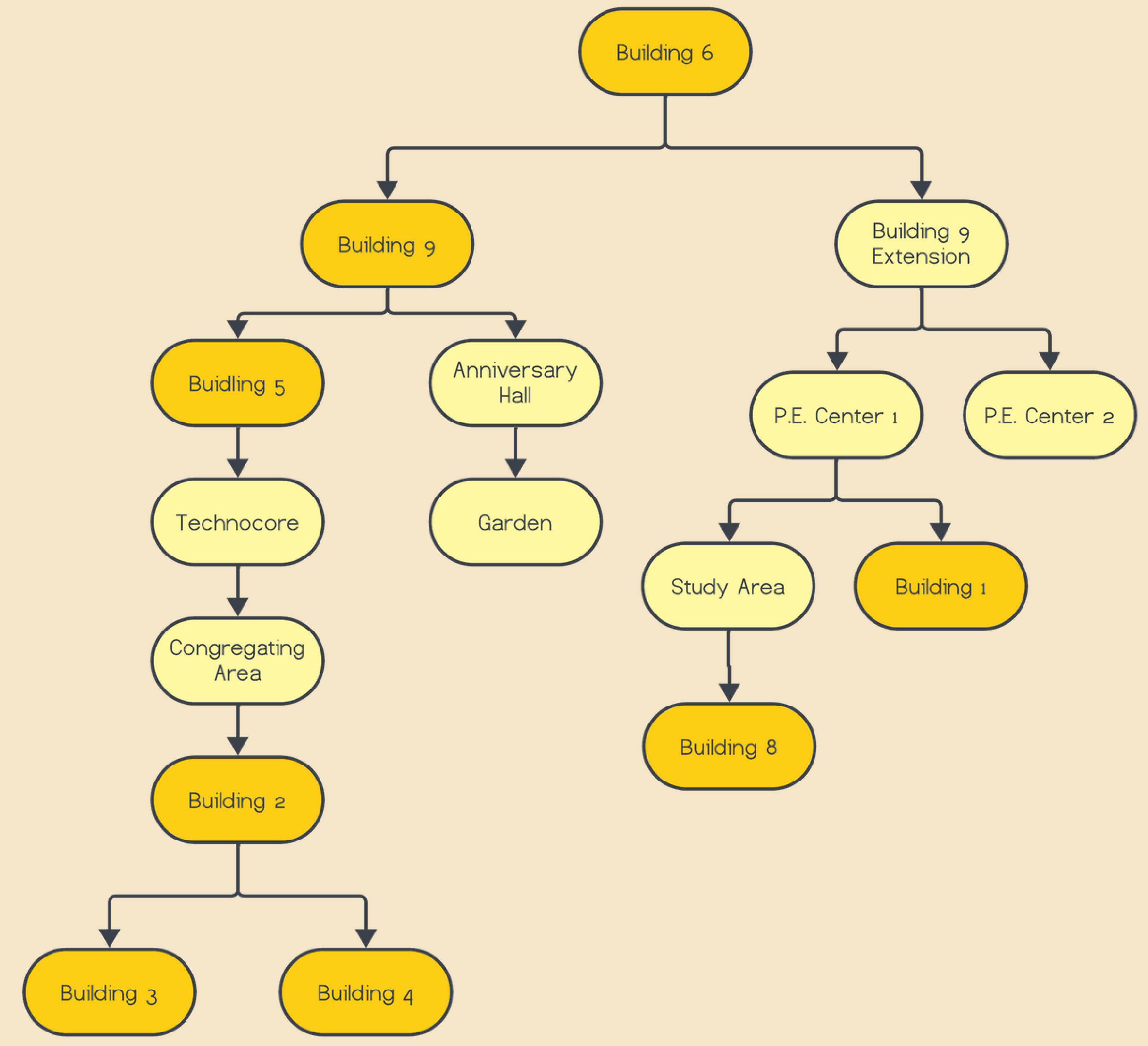
```
The shortest path from Building 6 to Building 3 is Building 6 -> Building 9 -> Building 5 -> Technocore -> Congregating Area ->
Building 2 -> Building 3
```

[8]:

```python
shortest_bfs_path = BFS(graph, node14, node1)
if shortest_bfs_path:
    path = " -> ".join([node.getName() for node in shortest_bfs_path])
    print(f"The shortest path from {shortest_bfs_path[0].getName()} to {shortest_bfs_path[-1].getName()} is {path}")
else:
    print("No path exists between the nodes.")
```

```
The shortest path from Congregating Area to Building 1 is Congregating Area -> Technocore -> Building 5 -> Building 9 -> Buildin
g 6 -> Building 9 Extension -> P.E. Center 1 -> Building 1
```

[ ]:


Building 6
Building 9
Building 9 Extension
Buidling 5
Anniversary Hall
P.E. Center 1
P.E. Center 2
Technocore
Garden
Study Area
Building 1
Congregating Area
Building 8
Building 2
Building 3
Building 4

THANK YOU!