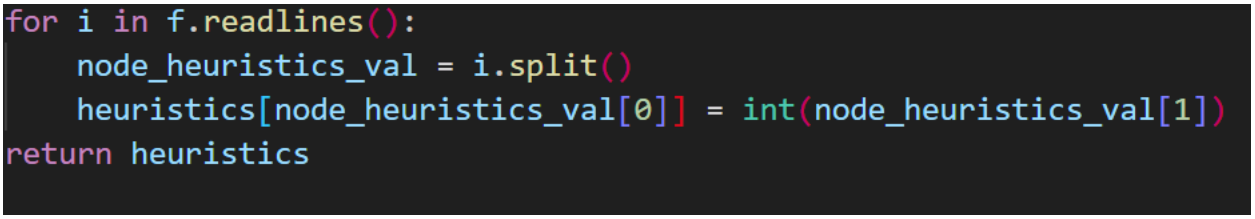
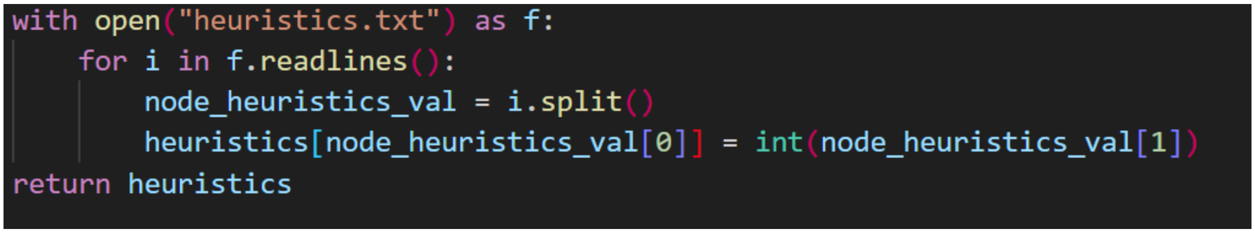
Problem in the code:

1/ In the function getHeuristics():



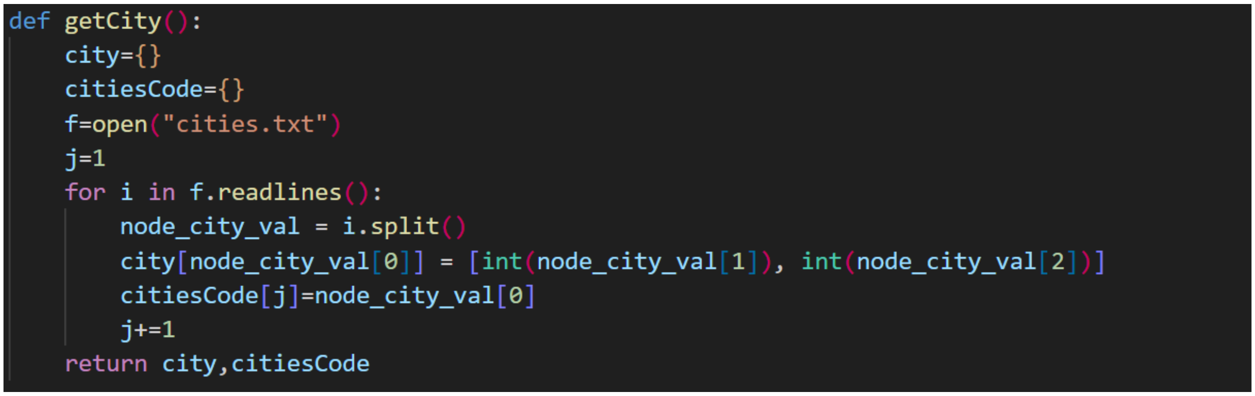
```
for i in f.readlines():
    node_heuristics_val = i.split()
    heuristics[node_heuristics_val[0]] = int(node_heuristics_val[1])
return heuristics
```

Here is the fixed code:



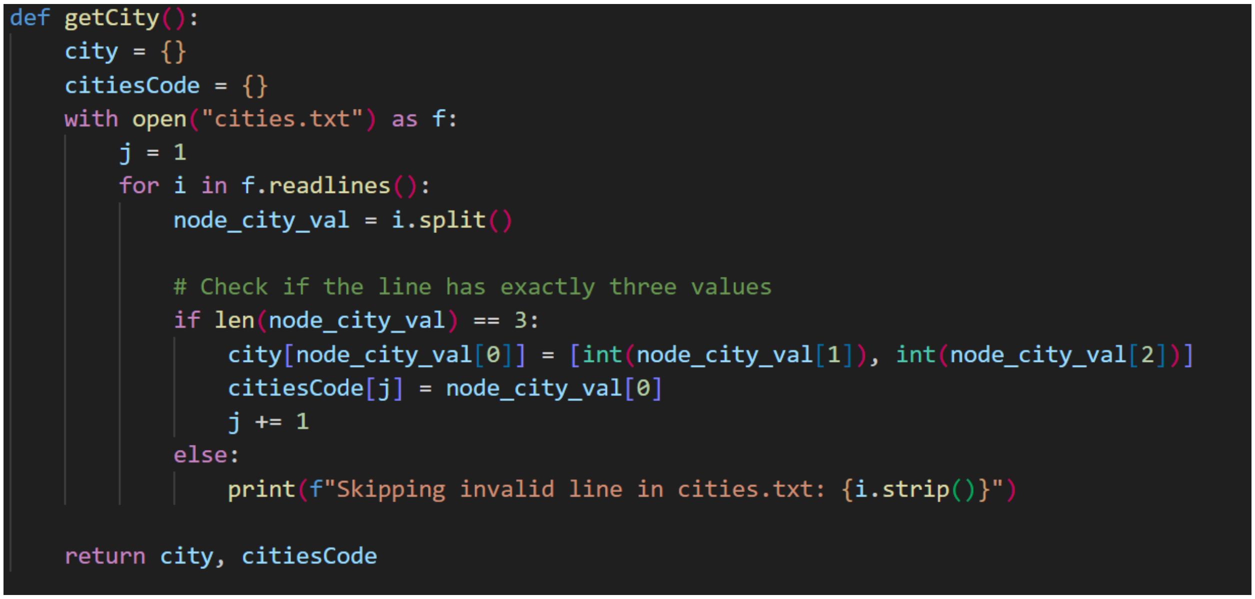
```
with open("heuristics.txt") as f:
    for i in f.readlines():
        node_heuristics_val = i.split()
        heuristics[node_heuristics_val[0]] = int(node_heuristics_val[1])
return heuristics
```

2/The code in function getCity() can easy bug, so we can write it a brand new code for this function:



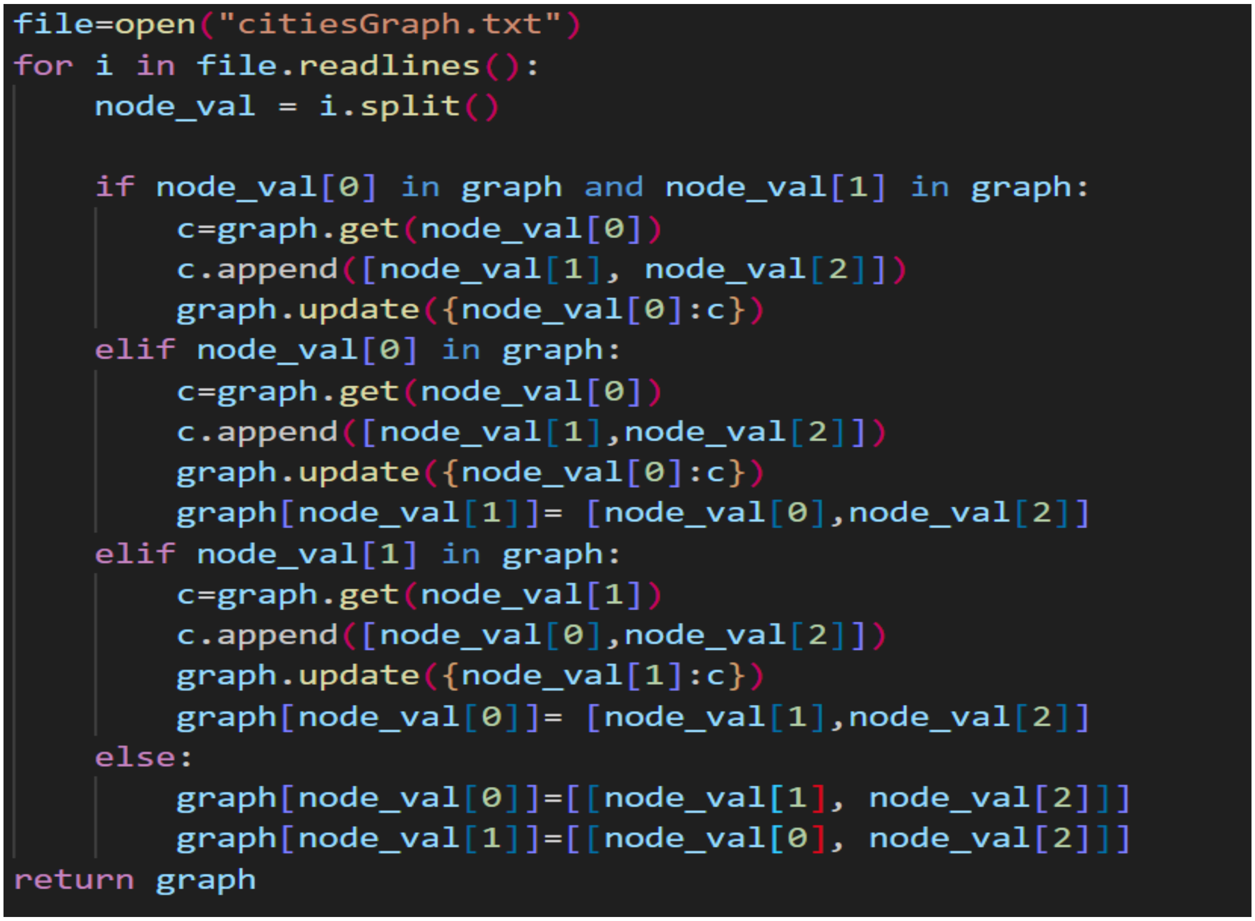
```
def getCity():
    city={}
    citiesCode={}
    f=open("cities.txt")
    j=1
    for i in f.readlines():
        node_city_val = i.split()
        city[node_city_val[0]] = [int(node_city_val[1]), int(node_city_val[2])]
        citiesCode[j]=node_city_val[0]
        j+=1
    return city,citiesCode
```

Here is the fixed code:



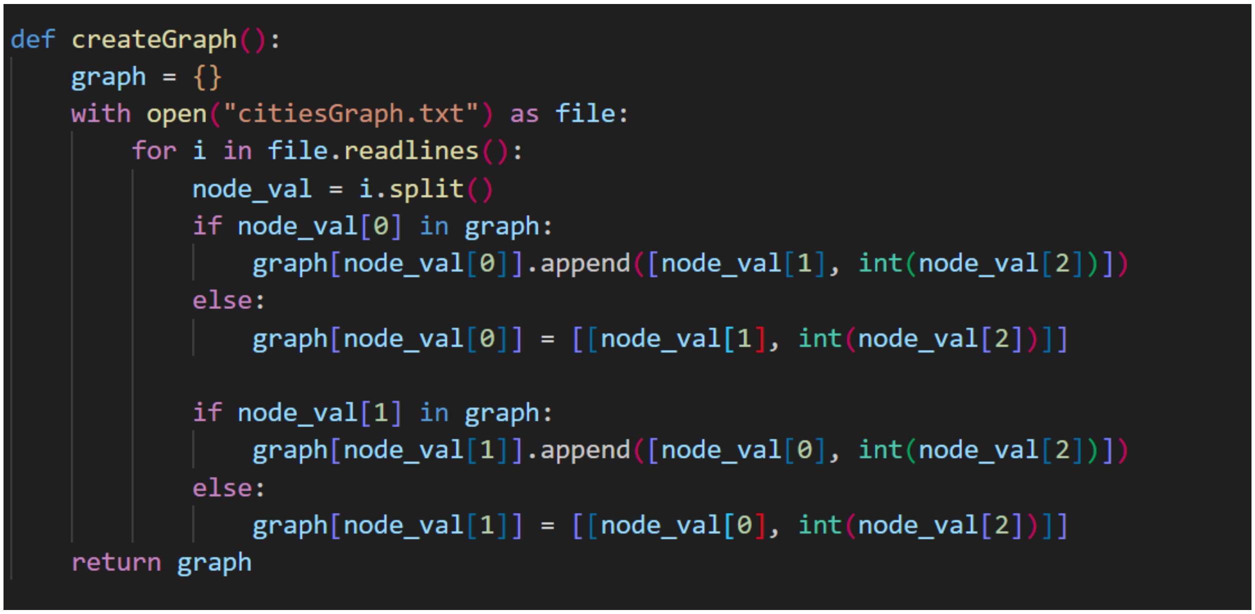
```python
def getCity():
    city = {}
    citiesCode = {}
    with open("cities.txt") as f:
        j = 1
        for i in f.readlines():
            node_city_val = i.split()

            # Check if the line has exactly three values
            if len(node_city_val) == 3:
                city[node_city_val[0]] = [int(node_city_val[1]), int(node_city_val[2])]
                citiesCode[j] = node_city_val[0]
                j += 1
            else:
                print(f"Skipping invalid line in cities.txt: {i.strip()}")

    return city, citiesCode
```

3/In createGraph function, here is the problem:



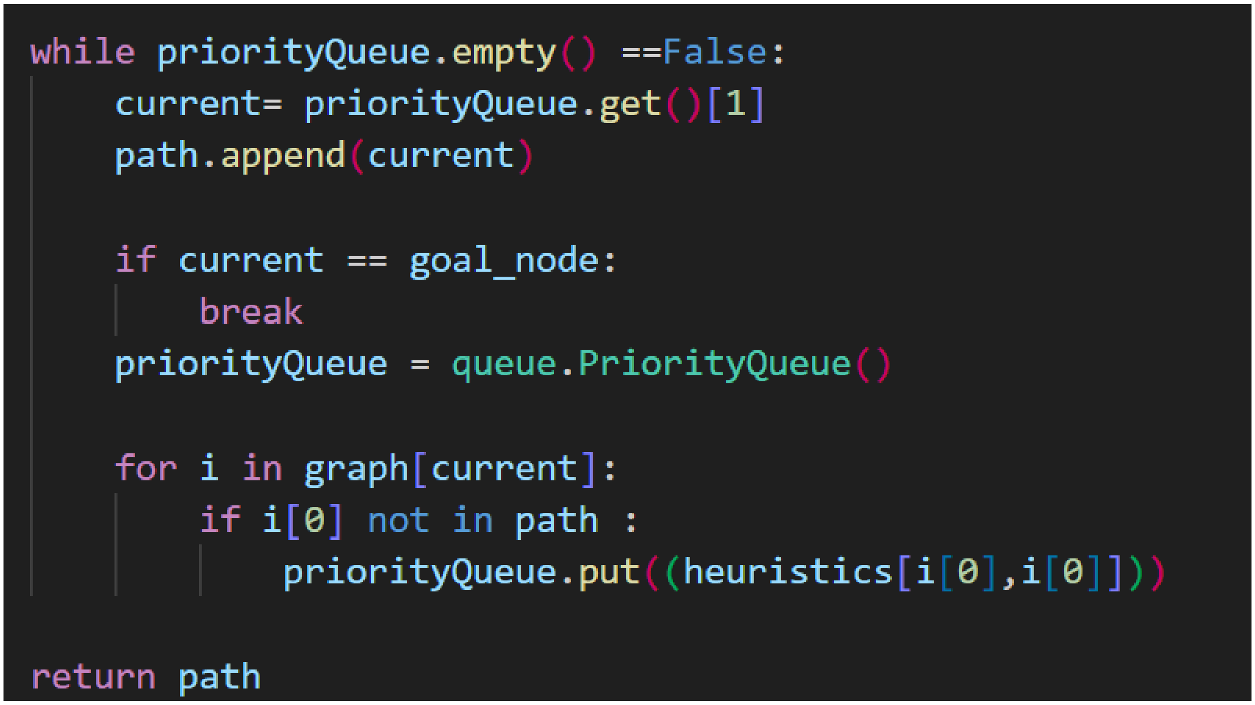
```
file=open("citiesGraph.txt")
for i in file.readlines():
    node_val = i.split()

    if node_val[0] in graph and node_val[1] in graph:
        c=graph.get(node_val[0])
        c.append([node_val[1], node_val[2]])
        graph.update({node_val[0]:c})
    elif node_val[0] in graph:
        c=graph.get(node_val[0])
        c.append([node_val[1],node_val[2]])
        graph.update({node_val[0]:c})
        graph[node_val[1]]= [node_val[0],node_val[2]]
    elif node_val[1] in graph:
        c=graph.get(node_val[1])
        c.append([node_val[0],node_val[2]])
        graph.update({node_val[1]:c})
        graph[node_val[0]]= [node_val[1],node_val[2]]
    else:
        graph[node_val[0]]=[[node_val[1], node_val[2]]]
        graph[node_val[1]]=[[node_val[0], node_val[2]]]
return graph
```

Here is the fixed code:



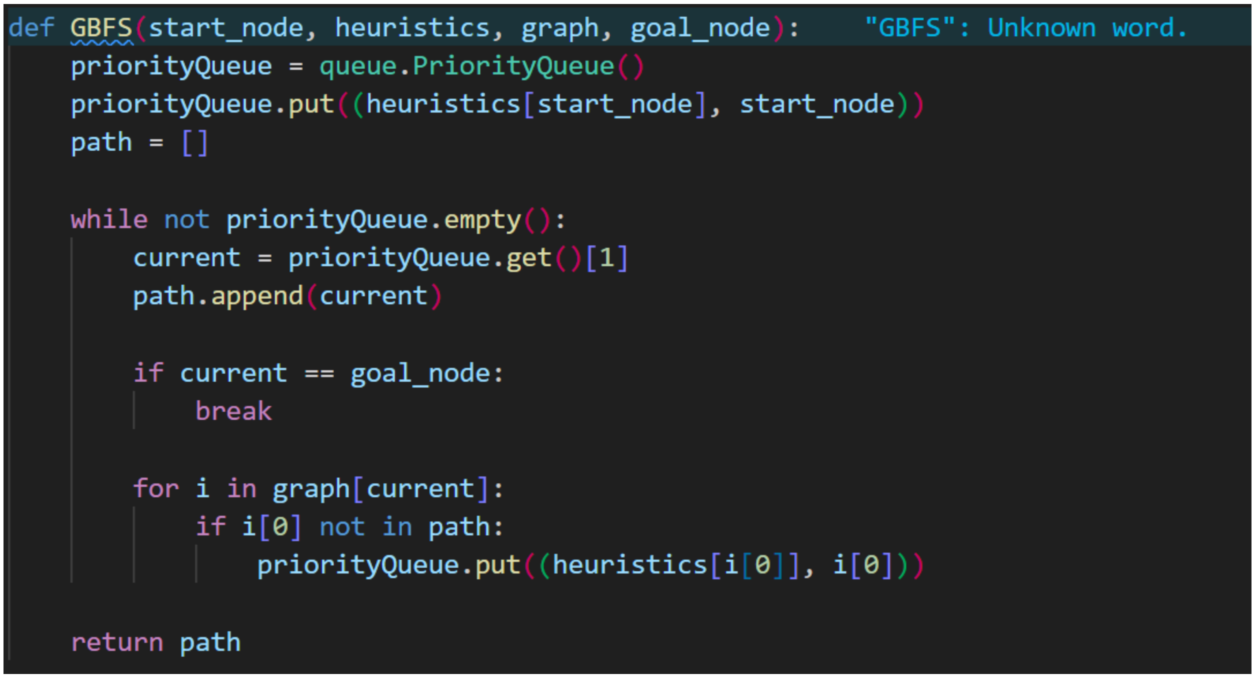
```
def createGraph():
    graph = {}
    with open("citiesGraph.txt") as file:
        for i in file.readlines():
            node_val = i.split()
            if node_val[0] in graph:
                graph[node_val[0]].append([node_val[1], int(node_val[2])])
            else:
                graph[node_val[0]] = [[node_val[1], int(node_val[2])]]

            if node_val[1] in graph:
                graph[node_val[1]].append([node_val[0], int(node_val[2])])
            else:
                graph[node_val[1]] = [[node_val[0], int(node_val[2])]]
    return graph
```

4/In function GBFS(), we just need to delete the line “priorityQueue = queue.PriorityQueue()”



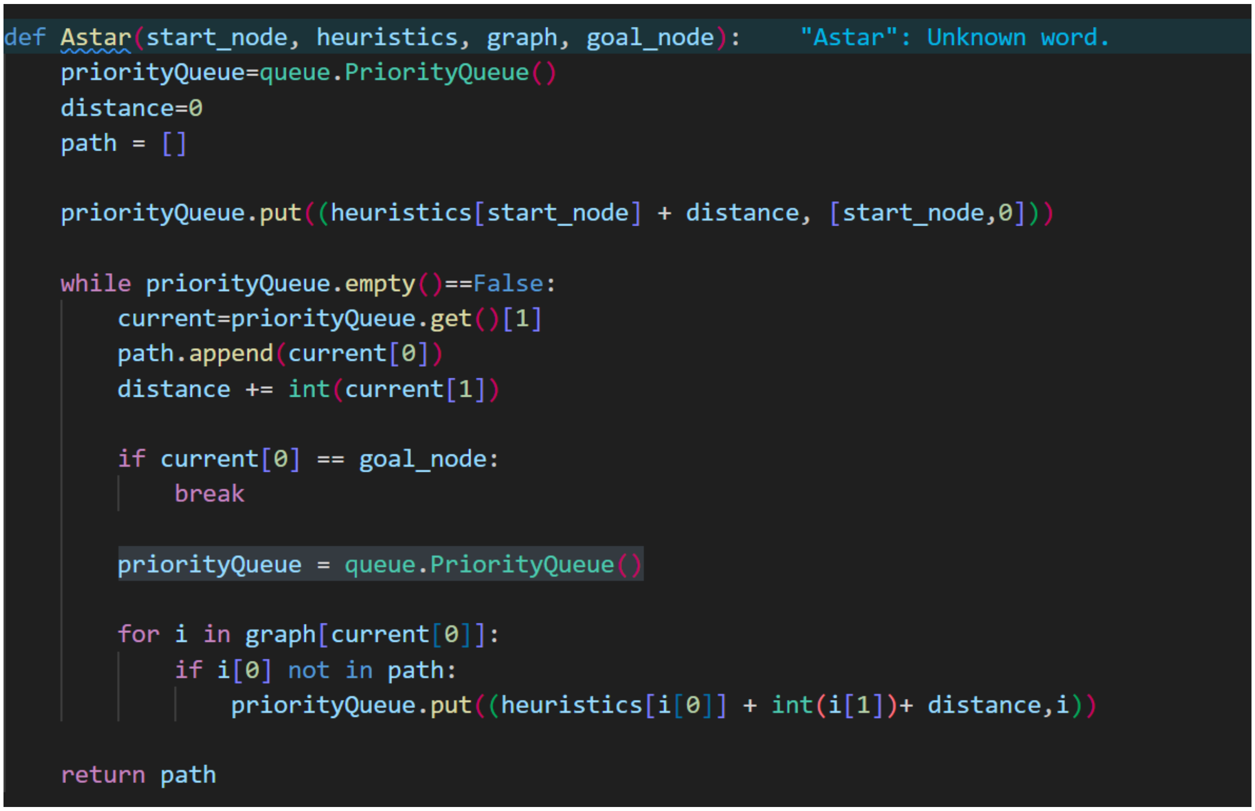
```
while priorityQueue.empty() ==False:
    current= priorityQueue.get()[1]
    path.append(current)

    if current == goal_node:
        break
    priorityQueue = queue.PriorityQueue()

    for i in graph[current]:
        if i[0] not in path :
            priorityQueue.put((heuristics[i[0],i[0]]))

return path
```

Here is the fixed code:



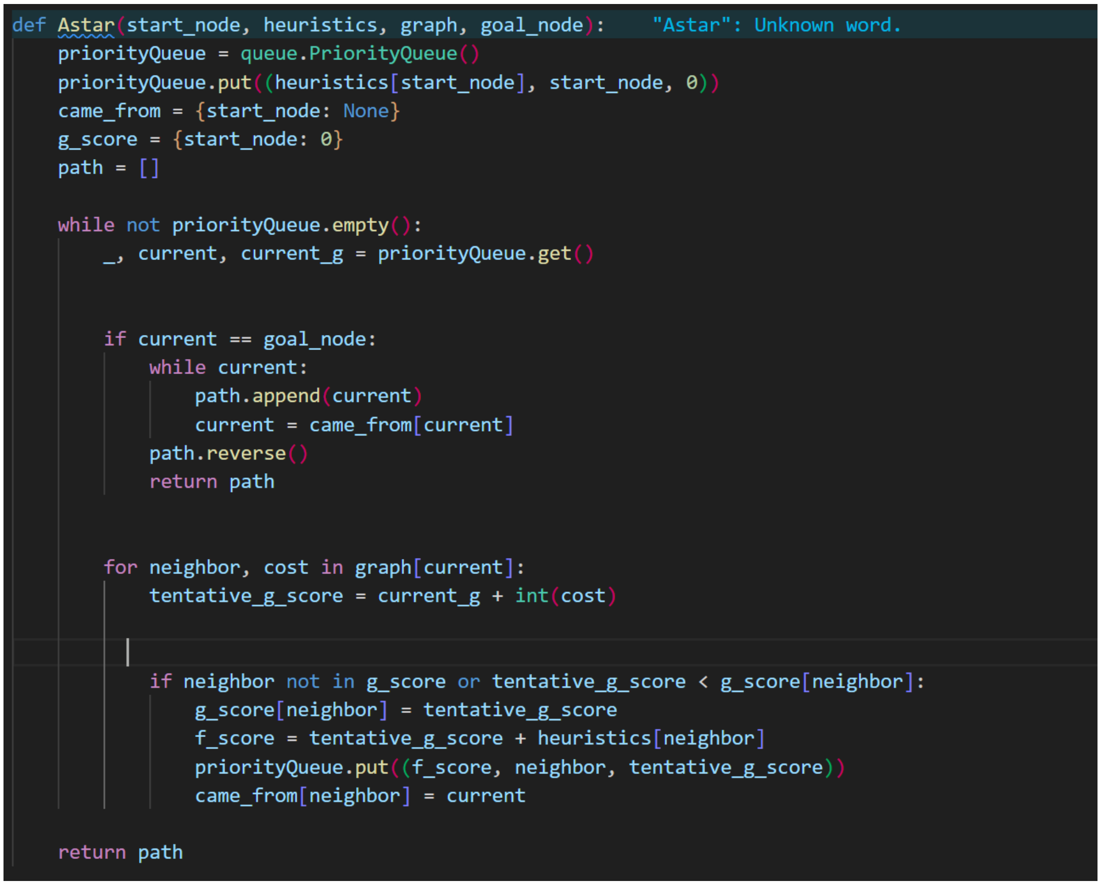
```
def GBFS(start_node, heuristics, graph, goal_node):    "GBFS": Unknown word.
    priorityQueue = queue.PriorityQueue()
    priorityQueue.put((heuristics[start_node], start_node))
    path = []

    while not priorityQueue.empty():
        current = priorityQueue.get()[1]
        path.append(current)

        if current == goal_node:
            break

        for i in graph[current]:
            if i[0] not in path:
                priorityQueue.put((heuristics[i[0]], i[0]))

    return path
```

5/In Astar() function, there is many issue so I will write it again, here is the origin code:



```
def Astar(start_node, heuristics, graph, goal_node):    "Astar": Unknown word.
    priorityQueue=queue.PriorityQueue()
    distance=0
    path = []

    priorityQueue.put((heuristics[start_node] + distance, [start_node,0]))

    while priorityQueue.empty()==False:
        current=priorityQueue.get()[1]
        path.append(current[0])
        distance += int(current[1])

        if current[0] == goal_node:
            break

        priorityQueue = queue.PriorityQueue()

        for i in graph[current[0]]:
            if i[0] not in path:
                priorityQueue.put((heuristics[i[0]] + int(i[1])+ distance,i))

    return path
```

Here is the fixed code:



```
def Astar(start_node, heuristics, graph, goal_node):     "Astar": Unknown word.
    priorityQueue = queue.PriorityQueue()
    priorityQueue.put((heuristics[start_node], start_node, 0))
    came_from = {start_node: None}
    g_score = {start_node: 0}
    path = []

    while not priorityQueue.empty():
        _, current, current_g = priorityQueue.get()


        if current == goal_node:
            while current:
                path.append(current)
                current = came_from[current]
            path.reverse()
            return path


        for neighbor, cost in graph[current]:
            tentative_g_score = current_g + int(cost)


            if neighbor not in g_score or tentative_g_score < g_score[neighbor]:
                g_score[neighbor] = tentative_g_score
                f_score = tentative_g_score + heuristics[neighbor]
                priorityQueue.put((f_score, neighbor, tentative_g_score))
                came_from[neighbor] = current

    return path
```