Practical 1 a] BFS(Breadth First Search)

Code:  
def bfs\_traverse(visited, graph, start):

visited.append(start)

queue.append(start)

while queue:

m = queue.pop(0)

print(m)

for neighbour in graph[m]:

if neighbour not in visited:

visited.append(neighbour)

queue.append(neighbour)

if \_\_name\_\_ == "\_\_main\_\_":

Graph = {

"Charni Road": ["Marine Drive", "Haji Ali"],

"Marine Drive": ["Charni Road", "Nariman Point"],

"Nariman Point": ["Marine Drive", "CST"],

"CST": ["Nariman Point", "Parel"],

"Parel": ["CST", "Kurla"],

"Kurla": ["Parel", "BKC"],

"BKC": ["Kurla", "Lower Parel"],

"Haji Ali": ["Charni Road", "Mahalaxmi"],

"Mahalaxmi": ["Haji Ali", "Worli"],

"Worli": ["Mahalaxmi", "Lower Parel"],

"Lower Parel": ["Worli", "BKC"]

}

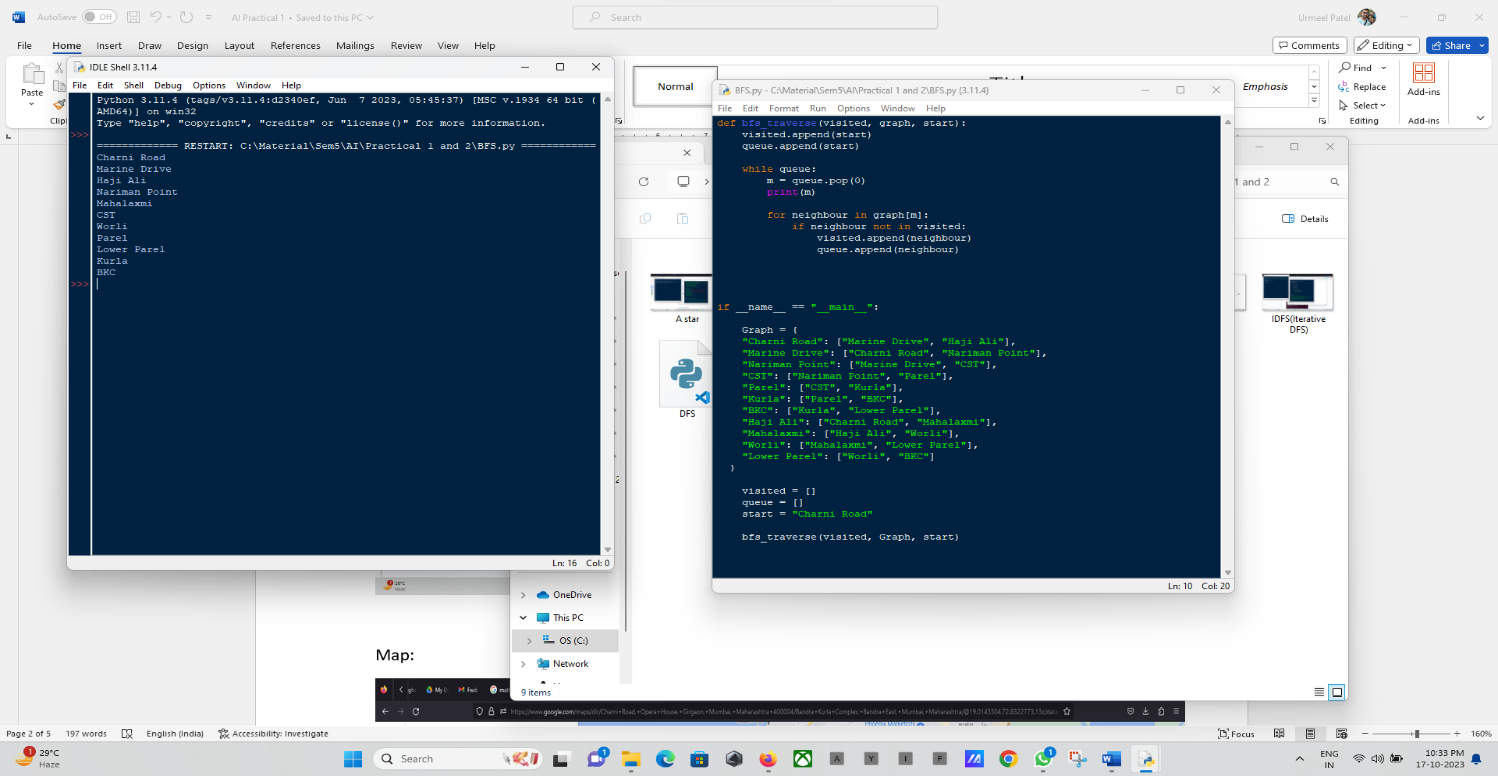
visited = []

queue = []

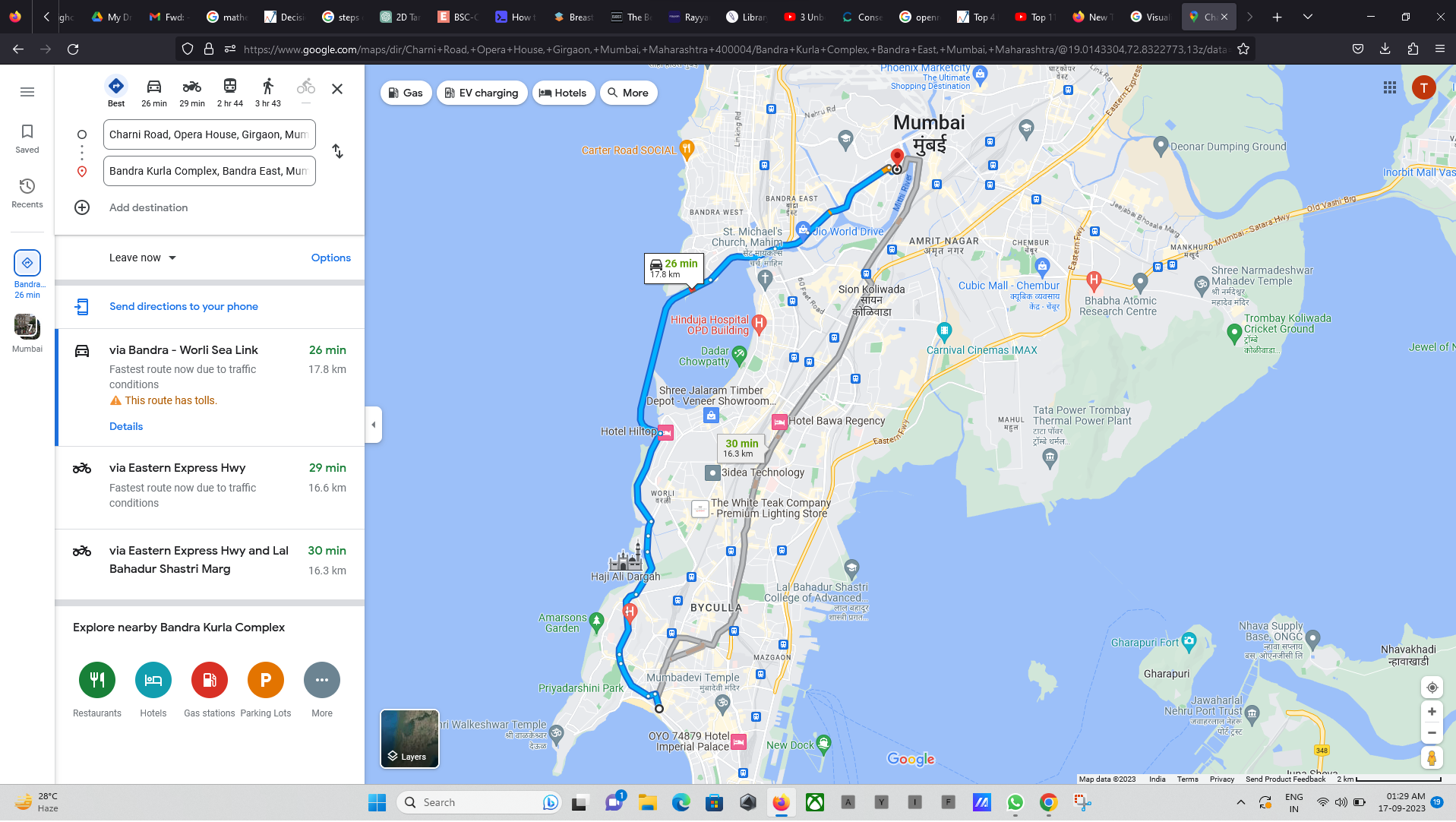
start = "Charni Road"

bfs\_traverse(visited, Graph, start)

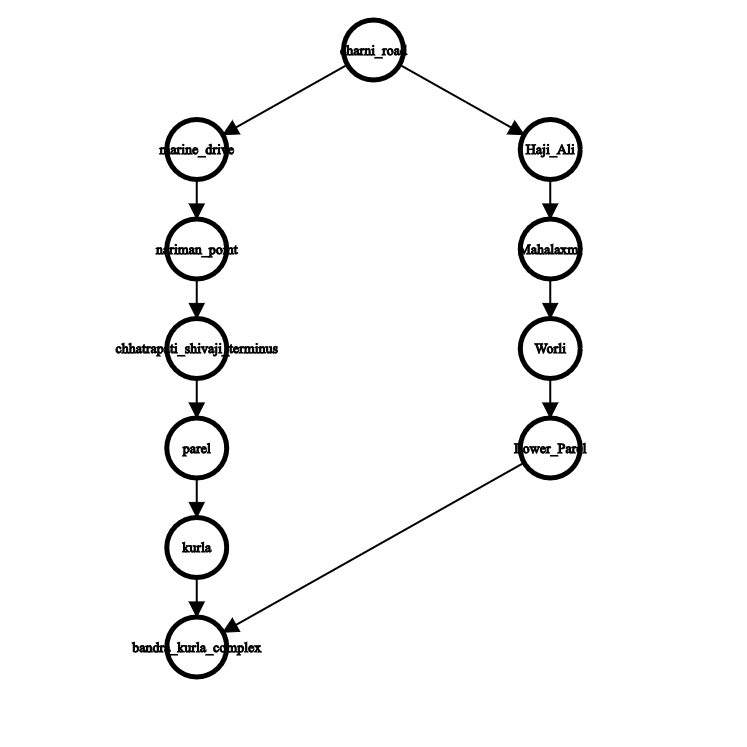
Output:



Map:



Graph:



b] IDFS(Iterative Depth First Search)

Code:

from collections import deque

def iterativeDFS(graph, start, destination):

stack = deque()

visited = []

stack.appendleft(start)

while stack:

node = stack.popleft()

if node in visited:

continue

visited.append(node)

print(node)

if node == destination:

return

for neighbor in graph[node]:

if neighbor not in visited:

stack.appendleft(neighbor)

if \_\_name\_\_ == "\_\_main\_\_":

Graph = {

"Charni Road": ["Marine Drive", "Haji Ali"],

"Marine Drive": ["Charni Road", "Nariman Point"],

"Nariman Point": ["Marine Drive", "CST"],

"CST": ["Nariman Point", "Parel"],

"Parel": ["CST", "Kurla"],

"Kurla": ["Parel", "BKC"],

"BKC": ["Kurla", "Lower Parel"],

"Haji Ali": ["Charni Road", "Mahalaxmi"],

"Mahalaxmi": ["Haji Ali", "Worli"],

"Worli": ["Mahalaxmi", "Lower Parel"],

"Lower Parel": ["Worli", "BKC"],

"BKC": []

}

start = "Charni Road"

destination = "BKC"

iterativeDFS(Graph, start, destination)

Output:

