**Task 01:**

f1 = open("20101539\_Umme Abira Azmary\_CSE422\_11\_Assignment01\_Summer2024\_InputFile.txt", "r")

f2 = open("output.txt", "w")

import heapq

class MinimumPQueue:

def \_\_init\_\_(self):

self.varcount = 0

self.priorty\_list = []

def nodepush(self, tobject, priorty\_value):

heapq.heappush(self.priorty\_list, (priorty\_value,self.varcount , tobject))

self.varcount += 1

def nodepop(self):

temp\_lst = heapq.heappop(self.priorty\_list)

return temp\_lst[-1]

def is\_empty(self):

temp\_lst = self.priorty\_list

if len(temp\_lst) == 0:

return True

else:

return False

class Node:

def \_\_init\_\_(self,name,pcost,hcost,pnode):

self.name = name

self.pcost = pcost

self.hcost = hcost

self.pnode = pnode

self.total\_cost = hcost+ pcost

pq = MinimumPQueue()

heurestic = {}

path\_costs = {}

expired\_parents = {}

for line in f1:

line = line.split()

main\_node = line[0]

heurestic[main\_node] = int(line[1])

path\_costs[main\_node] = []

for selm in range(2,len(line),2):

#print(line[selm],line[selm+1])

path\_costs[main\_node].append([line[selm], int(line[selm+1])])

#f2.write(line)

starting\_node = input("Enter starting node: ")

ending\_node = input("Enter destination node: ")

total\_path\_costs = 0

final\_node = None

node1 = Node(starting\_node,0,heurestic[starting\_node],None)

pq.nodepush(node1,node1.total\_cost)

while pq.is\_empty() == False:

node1 = pq.nodepop()

# print(node1.name)

noneflag = node1.pnode is None

if node1.name == ending\_node:

final\_node = node1

break

if node1.name not in expired\_parents.keys():

if node1.pnode:

expired\_parents[node1.name] = [node1.pnode.name]

else:

if node1.pnode:

if node1.pnode.name in expired\_parents[node1.name]:

continue

else:

expired\_parents[node1.name].append(node1.pnode.name)

for elm in path\_costs[node1.name]:

if noneflag or node1.pnode.name != elm[0]:

pcost = node1.pcost + elm[1]

new\_node = Node(elm[0],pcost,heurestic[elm[0]],node1)

pq.nodepush(new\_node,new\_node.total\_cost)

#print(heurestic)

#print(path\_costs)

temp\_node = final\_node

flst = []

while temp\_node is not None:

flst.append(temp\_node.name)

temp\_node = temp\_node.pnode

flst = flst[::-1]

if len(flst) >0:

# print(flst)

wordlst = ""

for selm in flst:

wordlst += selm + " -> "

wordlst = wordlst.rstrip("-> ")

print(f"Path: {wordlst}")

print(f"Total distance: {final\_node.pcost} km")

else:

print("NO PATH FOUND")

f1.close()

f2.close()