PROGRAM

from queue import PriorityQueue

class State(object):

def \_\_init\_\_(self, value, parent, start = 0, goal = 0):

self.children = []

self.parent = parent

self.value = value

self.dist = 0

if parent:

self.start = parent.start

self.goal = parent.goal

self.path = parent.path[:]

self.path.append(value)

else:

self.path = [value]

self.start = start

self.goal = goal

def GetDistance(self):

pass

def CreateChildren(self):

pass

class State\_String(State):

def \_\_init\_\_(self, value, parent, start = 0, goal = 0 ):

super(State\_String, self).\_\_init\_\_(value, parent, start, goal)

self.dist = self.GetDistance()

def GetDistance(self):

if self.value == self.goal:

return 0

dist = 0

for i in range(len(self.goal)):

letter = self.goal[i]

dist += abs(i - self.value.index(letter))

return dist

def CreateChildren(self):

if not self.children:

for i in range(len(self.goal)-1):

val = self.value

val = val[:i] + val[i+1] + val[i] + val[i+2:]

child = State\_String(val, self)

self.children.append(child)

class A\_Star\_Solver:

def \_\_init\_\_(self, start, goal):

self.path = []

self.vistedQueue =[]

self.priorityQueue = PriorityQueue()

self.start = start

self.goal = goal

def Solve(self):

startState = State\_String(self.start,0,self.start,self.goal)

count = 0

self.priorityQueue.put((0,count, startState))

while(not self.path and self.priorityQueue.qsize()):

closesetChild = self.priorityQueue.get()[2]

closesetChild.CreateChildren()

self.vistedQueue.append(closesetChild.value)

for child in closesetChild.children:

if child.value not in self.vistedQueue:

count += 1

if not child.dist:

self.path = child.path

break

self.priorityQueue.put((child.dist,count,child))

if not self.path:

print("Goal Of is not possible !" + self.goal )

return self.path

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

start1 = "teja"

goal1 = "jate"

print("Starting....")

a = A\_Star\_Solver(start1,goal1)

a.Solve()

for i in range(len(a.path)):

print("{0}){1}".format(i,a.path[i]))

OUTPUT

