Pseudo Code For Breadth First Search

R,C- no.of rows and columns

m- i/p char matrix of size R\*C

sr,sc-Row and column value of source

rq,cq- empty row queue(rq)

empty column queue(cq)

#variable used to track steps taken – move\_count=0

nodes\_left\_in\_layer=1

nodes\_in\_next\_layer=0

#variable used to track whether char ever gets reached- reached\_end=false

#R\*C matrix of false values used to track whether the node at position (i,j) has been visited- visited(matrix)

#NSEW

dr=[-1,1,0,0]

dc=[0,0,1,-1]

function solve() :

rq.enqueue(sr)

cq.enqueue(sc)

visited[sr][sc] = true

while rq.size()>0 :

r= rq.dequeue()

c= cq.dequeue()

if m[r][c]== e :

reached\_end= true

break

function explore neighbours(r,c):

for(i=0;i<r;i++)

rr=r+dr

cc=c+dc

#skip out of boundary locations

if rr<0 or cc<0 : continue

if rr>=R or cc>=C : continue

#skip visited locations or blocked cells

if visited[rr][cc]: continue

if m[rr][cc] == “#” : continue

rq.enqueue(rr)

cq.enqueue(cc)

visited[rr][cc]=true

nodes\_in\_next\_layer++

nodes\_left\_in\_layer - -

if nodes\_left\_in\_layer==0:

nodes\_left\_in\_layer = nodes\_in\_next\_layer = 0

nodes\_in\_next\_layer=0

move\_count++

if reached\_end:

return move\_count

return -1