**TASK 1:**

**Implement A\* search and find the optimal path for finding the goal.**

**CODE:**

*# Owned*

\_\_author\_\_ *=* "Qaiser Abbas"

\_\_copyright\_\_ *=* "Copyright 2020, Artificial Intelligence lab-06"

\_\_email\_\_ *=* "qaiserabbas889@yahoo.com"

*#===============================================================*

*# {code}*

*import* queue *as* Q

*def* *search*(graph, start, end):

    whileiterations *=* 0

    foriteration *=* 0

*if* start *not* *in* graph:

*raise* TypeError(str(start) *+* ' not found in graph !')

*if* end *not* *in* graph:

*raise* TypeError(str(end) *+* ' not found in graph !')

    queue *=* Q.PriorityQueue()

    queue.put((0, [start]))

*while* *not* queue.empty():

        whileiterations *=* whileiterations*+*1

        node *=* queue.get()

        current *=* node[1][len(node[1]) *-* 1]

*if* end *in* node[1]:

            print("Path found: " *+* str(node[1]) *+* ", Cost = " *+* str(node[0]))

*break*

        cost *=* node[0]

*for* neighbor *in* graph[current]:

            foriteration *=* foriteration*+*1

            temp *=* node[1][:]

            temp.append(neighbor)

            queue.put((cost *+* graph[current][neighbor], temp))

    print("Total while loop executed "*+*str(whileiterations)*+*" times")

    print("Total for loop executed "*+*str(foriteration)*+*" times")

*def* *main*():

    graph *=* {

    'Bahria University': {'Gulshan e Iqbal': 5, 'Bahadurabad': 8, 'Shah Faisal': 4},

    'Gulshan e Iqbal': {'Gulistan e johar': 71, 'Bahria University': 35},

    'Bahadurabad': {'Bahria University': 18, 'NIPA': 11},

    'Shah Faisal': {'Bahria University': 90, 'Gulistan e johar': 31, 'Malir': 99, 'Airport': 60},

    'Gulistan e johar': {'Gulshan e Iqbal': 71, 'Shah Faisal': 51},

    'NIPA': {'Bahadurabad': 11, 'DHA': 70},

    'Airport': {'Shah Faisal': 80, 'Lyari': 97, 'Nazimabad': 46},

    'DHA': {'NIPA': 70, 'Karachi University': 75},

    'Nazimabad': {'Karachi University': 120, 'Airport': 146, 'Lyari': 138},

    'Lyari': {'Airport': 97, 'Nazimabad': 138, 'Korangi': 101},

    'Malir': {'Shah Faisal': 99, 'Korangi': 211},

    'Karachi University': {'DHA': 75, 'Nazimabad': 120},

    'Korangi': {'Malir': 211, 'Lyari': 101, 'Saddar': 90},

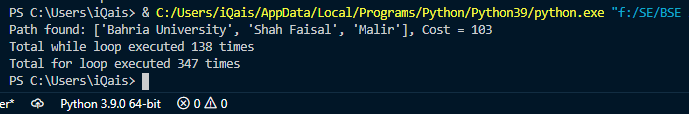
    'Saddar': {'Korangi': 90}

    }

    search(graph, 'Bahria University', 'Malir')

main()

**OUTPUT:**



**TASK 2:**

**Apply A\* search on pacman game**

**CODE:**

*# Owned*

\_\_author\_\_ *=* "Qaiser Abbas"

\_\_copyright\_\_ *=* "Copyright 2020, Artificial Intelligence lab-06"

\_\_email\_\_ *=* "qaiserabbas889@yahoo.com"

*#===============================================================*

*# {code}*

*import* pygame

*import* sys

*import* copy

*from* settings *import* *\**

*from* player\_class *import* *\**

*from* enemy\_class *import* *\**

pygame.init()

vec *=* pygame.math.Vector2

*class* App:

*def* \_\_init\_\_(self):

*self*.screen *=* pygame.display.set\_mode((WIDTH, HEIGHT))

*self*.clock *=* pygame.time.Clock()

*self*.running *=* True

*self*.state *=* 'start'

*self*.cell\_width *=* MAZE\_WIDTH*//*COLS

*self*.cell\_height *=* MAZE\_HEIGHT*//*ROWS

*self*.walls *=* []

*self*.coins *=* []

*self*.enemies *=* []

*self*.e\_pos *=* []

*self*.p\_pos *=* None

*self*.load()

*self*.player *=* Player(*self*, vec(*self*.p\_pos))

*self*.make\_enemies()

*def* *run*(self):

*while* *self*.running:

*if* *self*.state *==* 'start':

*self*.start\_events()

*self*.start\_update()

*self*.start\_draw()

*elif* *self*.state *==* 'playing':

*self*.playing\_events()

*self*.playing\_update()

*self*.playing\_draw()

*elif* *self*.state *==* 'game over':

*self*.game\_over\_events()

*self*.game\_over\_update()

*self*.game\_over\_draw()

*else*:

*self*.running *=* False

*self*.clock.tick(FPS)

        pygame.quit()

        sys.exit()

*############################ HELPER FUNCTIONS ##################################*

*def* *draw\_text*(self, words, screen, pos, size, colour, font\_name, centered*=*False):

        font *=* pygame.font.SysFont(font\_name, size)

        text *=* font.render(words, False, colour)

        text\_size *=* text.get\_size()

*if* centered:

            pos[0] *=* pos[0]*-*text\_size[0]*//*2

            pos[1] *=* pos[1]*-*text\_size[1]*//*2

        screen.blit(text, pos)

*def* *load*(self):

*self*.background *=* pygame.image.load('maze.png')

*self*.background *=* pygame.transform.scale(*self*.background, (MAZE\_WIDTH, MAZE\_HEIGHT))

*# Opening walls file*

*# Creating walls list with co-ords of walls*

*# stored as  a vector*

*with* open("walls.txt", 'r') *as* *file*:

*for* yidx, line *in* enumerate(*file*):

*for* xidx, char *in* enumerate(line):

*if* char *==* "1":

*self*.walls.append(vec(xidx, yidx))

*elif* char *==* "C":

*self*.coins.append(vec(xidx, yidx))

*elif* char *==* "P":

*self*.p\_pos *=* [xidx, yidx]

*elif* char *in* ["2", "3", "4", "5"]:

*self*.e\_pos.append([xidx, yidx])

*elif* char *==* "B":

                        pygame.draw.rect(*self*.background, BLACK, (xidx*\*self*.cell\_width, yidx*\*self*.cell\_height,

*self*.cell\_width, *self*.cell\_height))

*def* *make\_enemies*(self):

*for* idx, pos *in* enumerate(*self*.e\_pos):

*self*.enemies.append(Enemy(*self*, vec(pos), idx))

*def* *draw\_grid*(self):

*for* x *in* range(WIDTH*//self*.cell\_width):

            pygame.draw.line(*self*.background, GREY, (x*\*self*.cell\_width, 0),

                             (x*\*self*.cell\_width, HEIGHT))

*for* x *in* range(HEIGHT*//self*.cell\_height):

            pygame.draw.line(*self*.background, GREY, (0, x*\*self*.cell\_height),

                             (WIDTH, x*\*self*.cell\_height))

*# for coin in self.coins:*

*#     pygame.draw.rect(self.background, (167, 179, 34), (coin.x\*self.cell\_width,*

*#                                                        coin.y\*self.cell\_height, self.cell\_width, self.cell\_height))*

*def* *reset*(self):

*self*.player.lives *=* 3

*self*.player.current\_score *=* 0

*self*.player.grid\_pos *=* vec(*self*.player.starting\_pos)

*self*.player.pix\_pos *=* *self*.player.get\_pix\_pos()

*self*.player.direction *\*=* 0

*for* enemy *in* *self*.enemies:

            enemy.grid\_pos *=* vec(enemy.starting\_pos)

            enemy.pix\_pos *=* enemy.get\_pix\_pos()

            enemy.direction *\*=* 0

*self*.coins *=* []

*with* open("walls.txt", 'r') *as* *file*:

*for* yidx, line *in* enumerate(*file*):

*for* xidx, char *in* enumerate(line):

*if* char *==* 'C':

*self*.coins.append(vec(xidx, yidx))

*self*.state *=* "playing"

*########################### INTRO FUNCTIONS ####################################*

*def* *start\_events*(self):

*for* event *in* pygame.event.get():

*if* event.type *==* pygame.QUIT:

*self*.running *=* False

*if* event.type *==* pygame.KEYDOWN *and* event.key *==* pygame.K\_SPACE:

*self*.state *=* 'playing'

*def* *start\_update*(self):

*pass*

*def* *start\_draw*(self):

*self*.screen.fill(BLACK)

*self*.draw\_text('PUSH SPACE BAR', *self*.screen, [

                       WIDTH*//*2, HEIGHT*//*2*-*50], START\_TEXT\_SIZE, (170, 132, 58), START\_FONT, centered*=*True)

*self*.draw\_text('1 PLAYER ONLY', *self*.screen, [

                       WIDTH*//*2, HEIGHT*//*2*+*50], START\_TEXT\_SIZE, (44, 167, 198), START\_FONT, centered*=*True)

*self*.draw\_text('HIGH SCORE', *self*.screen, [4, 0],

                       START\_TEXT\_SIZE, (255, 255, 255), START\_FONT)

        pygame.display.update()

*########################### PLAYING FUNCTIONS ##################################*

*def* *playing\_events*(self):

*for* event *in* pygame.event.get():

*if* event.type *==* pygame.QUIT:

*self*.running *=* False

*if* event.type *==* pygame.KEYDOWN:

*if* event.key *==* pygame.K\_LEFT:

*self*.player.move(vec(*-*1, 0))

*if* event.key *==* pygame.K\_RIGHT:

*self*.player.move(vec(1, 0))

*if* event.key *==* pygame.K\_UP:

*self*.player.move(vec(0, *-*1))

*if* event.key *==* pygame.K\_DOWN:

*self*.player.move(vec(0, 1))

*def* *playing\_update*(self):

*self*.player.update()

*for* enemy *in* *self*.enemies:

            enemy.update()

*for* enemy *in* *self*.enemies:

*if* enemy.grid\_pos *==* *self*.player.grid\_pos:

*self*.remove\_life()

*def* *playing\_draw*(self):

*self*.screen.fill(BLACK)

*self*.screen.blit(*self*.background, (TOP\_BOTTOM\_BUFFER*//*2, TOP\_BOTTOM\_BUFFER*//*2))

*self*.draw\_coins()

*# self.draw\_grid()*

*self*.draw\_text('CURRENT SCORE: {}'.format(*self*.player.current\_score),

*self*.screen, [60, 0], 18, WHITE, START\_FONT)

*self*.draw\_text('HIGH SCORE: 0', *self*.screen, [WIDTH*//*2*+*60, 0], 18, WHITE, START\_FONT)

*self*.player.draw()

*for* enemy *in* *self*.enemies:

            enemy.draw()

        pygame.display.update()

*def* *remove\_life*(self):

*self*.player.lives *-=* 1

*if* *self*.player.lives *==* 0:

*self*.state *=* "game over"

*else*:

*self*.player.grid\_pos *=* vec(*self*.player.starting\_pos)

*self*.player.pix\_pos *=* *self*.player.get\_pix\_pos()

*self*.player.direction *\*=* 0

*for* enemy *in* *self*.enemies:

                enemy.grid\_pos *=* vec(enemy.starting\_pos)

                enemy.pix\_pos *=* enemy.get\_pix\_pos()

                enemy.direction *\*=* 0

*def* *draw\_coins*(self):

*for* coin *in* *self*.coins:

            pygame.draw.circle(*self*.screen, (124, 123, 7),

                               (int(coin.x*\*self*.cell\_width)*+self*.cell\_width*//*2*+*TOP\_BOTTOM\_BUFFER*//*2,

                                int(coin.y*\*self*.cell\_height)*+self*.cell\_height*//*2*+*TOP\_BOTTOM\_BUFFER*//*2), 5)

*########################### GAME OVER FUNCTIONS ################################*

*def* *game\_over\_events*(self):

*for* event *in* pygame.event.get():

*if* event.type *==* pygame.QUIT:

*self*.running *=* False

*if* event.type *==* pygame.KEYDOWN *and* event.key *==* pygame.K\_SPACE:

*self*.reset()

*if* event.type *==* pygame.KEYDOWN *and* event.key *==* pygame.K\_ESCAPE:

*self*.running *=* False

*def* *game\_over\_update*(self):

*pass*

*def* *game\_over\_draw*(self):

*self*.screen.fill(BLACK)

        quit\_text *=* "Press the escape button to QUIT"

        again\_text *=* "Press SPACE bar to PLAY AGAIN"

*self*.draw\_text("GAME OVER", *self*.screen, [WIDTH*//*2, 100],  52, RED, "arial", centered*=*True)

*self*.draw\_text(again\_text, *self*.screen, [

                       WIDTH*//*2, HEIGHT*//*2],  36, (190, 190, 190), "arial", centered*=*True)

*self*.draw\_text(quit\_text, *self*.screen, [

                       WIDTH*//*2, HEIGHT*//*1.5],  36, (190, 190, 190), "arial", centered*=*True)

        pygame.display.update()

**OUTPUT:**

