

Mid Assignment Report

CSE-0408 Summer 2021

Khondkar Md Mufrat tasif
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
kmmufrat08@gmail.com

Abstract—The 8-puzzle problem is a puzzle invented and popularized by Noyes Palmer Chapman in the 1870s. It is played on a 3-by-3 grid with 8 square blocks labeled 1 through 8 and a blank square. Your goal is to rearrange the blocks so that they are in order.

A Breadth-first search(BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key') and explores the neighbor nodes first, before moving to the next level neighbors.

Index Terms—C++ , python

I. INTRODUCTION

The puzzle can be solved by moving the tiles one by one in the single empty space and thus achieving the Goal state.

Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration

II. WHICH/WAT ALGORITHMS USE 8 PUZZLE AND BFS?

Algorithm is one of the best and popular techniques used for path finding and graph traversals. A lot of games and web-based maps use this algorithm for finding the shortest path efficiently. It is essentially a best first search algorithm.

BFS is also used in the famous Dijkstra's algorithm for computing the shortest path in a graph and the Ford-Fulkerson algorithm for computing the maximum flow in a flow network. Here is an example of a map that BFS can take and return the shortest paths.

III. WHAT IS THE 8-PUZZLE PROBLEM? AND BFS ALGORITHM PROBLEM ?

The 8-puzzle problem is a puzzle invented and popularized by Noyes Palmer Chapman in the 1870s. It is played on a 3-by-3 grid with 8 square blocks labeled 1 through 8 and a blank square. Your goal is to rearrange the blocks so that they are in order.

Each vertex or node in the graph is known. In case the vertex V is not accessed then add the vertex V into the BFS Queue. Start the BFS search, and after completion, Mark vertex V as visited. The BFS queue is still not empty, hence remove the vertex V of the graph from the queue.

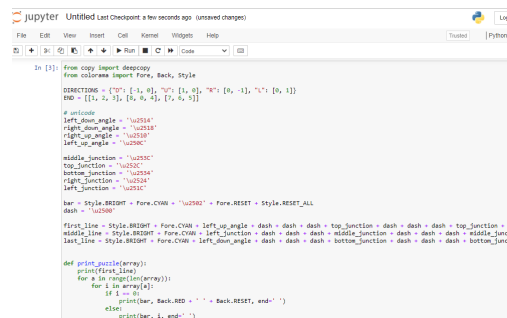
IV. RULES FOR SOLVING 8 PUZZLE

Instead of moving the tiles in the empty space we can visualize moving the empty space in place of the tile, basically swapping the tile with the empty space. The empty space can only move in four directions viz. 1. Up 2. Down 3. Right or 4. Left The empty space cannot move diagonally and can take only one step at a time.

V. RULES FOR SOLVING BFS ALGORITHM

Find the shortest path from source to destination in a matrix that satisfies given constraints. Find minimum passes required to convert all negative values in a matrix. Snake and Ladder Problem. Find the shortest distance of every cell from a landmine inside a maze.

VI. ASSIGNMENT CODE (8 PUZZLE PROBLEM)



```
from copy import deepcopy
from collections import deque, defaultdict
from sys import stdin, stdout

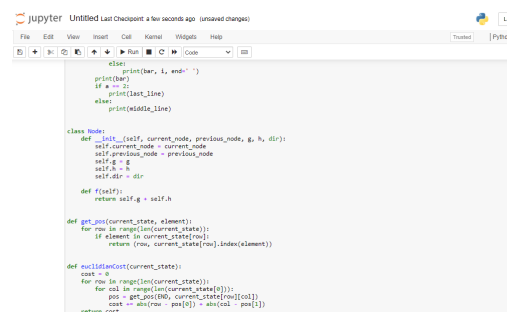
def get_pos(row, col):
    return row * 3 + col

def get_state(row, col):
    return row * 3 + col

def get_neighbours(state):
    row, col = divmod(state, 3)
    neighbours = []
    if row > 0:
        neighbours.append(get_pos(row-1, col))
    if row < 2:
        neighbours.append(get_pos(row+1, col))
    if col > 0:
        neighbours.append(get_pos(row, col-1))
    if col < 2:
        neighbours.append(get_pos(row, col+1))
    return neighbours

def bfs(start, goal):
    queue = deque([start])
    visited = set([start])
    while queue:
        state = queue.popleft()
        if state == goal:
            return state
        for neighbour in get_neighbours(state):
            if neighbour not in visited:
                queue.append(neighbour)
                visited.add(neighbour)
    return None
```

Fig. 1.



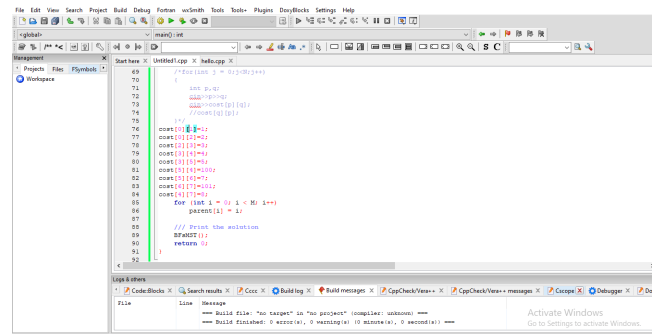
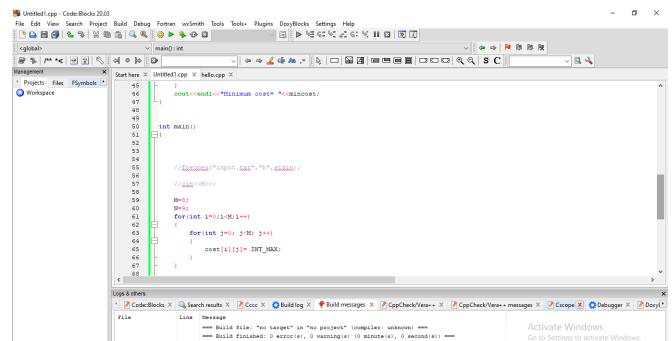
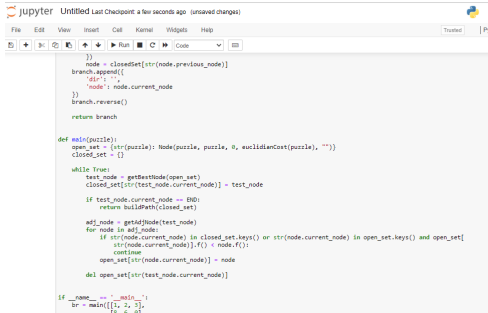
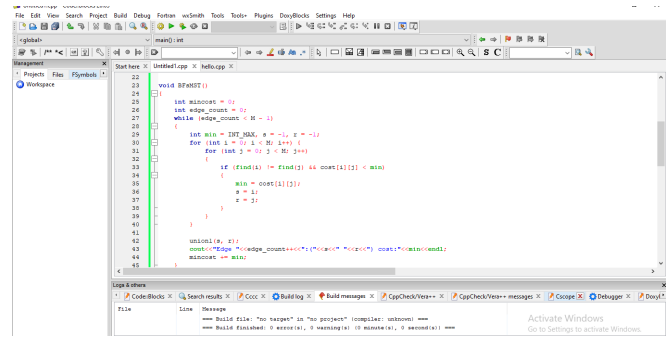
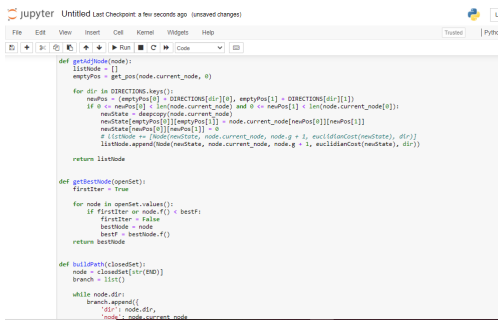
```
class Node:
    def __init__(self, current_node, previous_node, g, h, dir):
        self.current_node = current_node
        self.previous_node = previous_node
        self.g = g
        self.h = h
        self.dir = dir

    def __str__(self):
        return self.g + self.h

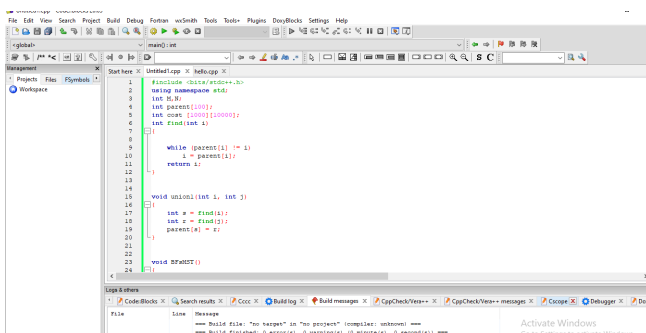
def get_pos(current_state, element):
    for row in range(3):
        for col in range(3):
            if element == current_state[row][col]:
                return (row, col)

def euclidean_cost(current_state):
    cost = 0
    for row in range(3):
        for col in range(3):
            pos = get_pos(8, current_state[row][col])
            cost += abs(row - pos[0]) + abs(col - pos[1])
    return cost
```

Fig. 2.



VII. ASSIGNMNET CODE(BFS ALGORITHM PROBLEM SOLVING)



VIII. ASSIGNMENT OUTPUT (8 PUZZLE PROBLEM AND BFS ALGORITHM PROBLEM SOLVING)



Fig. 10.

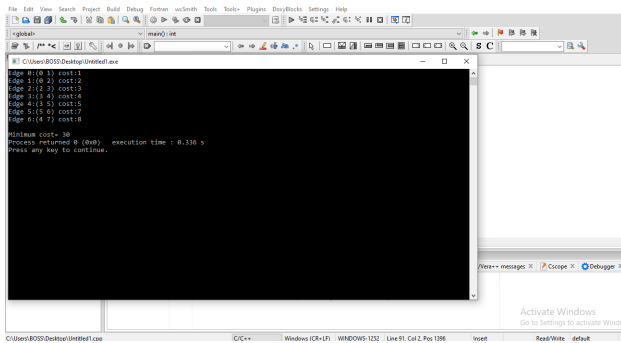


Fig. 11.

IX. CONCLUSION

i am testing my code to seeing that how many times it would take to get from the path, i am trying many of moves and sometimes works and sometimes not works .

The BFS algorithm is useful for analyzing the nodes in a graph and constructing the shortest path of traversing through these.

ACKNOWLEDGMENT

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

REFERENCES

- [1] Arboleya, P., Mohamed, B., González-Morán, C., El-Sayed, I. (2015). BFS algorithm for voltage-constrained meshed DC traction networks with nonsmooth voltage-dependent loads and generators. *IEEE Transactions on Power Systems*, 31(2), 1526-1536.
- [2] Chikkerur, S., Cartwright, A. N., Govindaraju, V. (2006, January). K-plet and coupled BFS: a graph based fingerprint representation and matching algorithm. In *International Conference on Biometrics* (pp. 309-315). Springer, Berlin, Heidelberg.
- [3] Gharehchopogh, F. S., Seyyedi, B., Feyzipour, G. (2012). a new solution for n-queens problem using blind approaches: DFS and BFS algorithms. *International Journal of Computer Applications*, 53(1).
- [4] Chakraborty, S., Satti, S. R. (2017, August). Space-efficient algorithms for maximum cardinality search, stack BFS, queue BFS and applications. In *International Computing and Combinatorics Conference* (pp. 87-98). Springer, Cham.
- [5] Awerbuch, B., Gallager, R. G. (1985, October). Distributed BFS algorithms. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)* (pp. 250-256). IEEE.
- [6] Reinefeld, A. (1993, August). Complete Solution of the Eight-Puzzle and the Bene t of Node Ordering in IDA*. In *International Joint Conference on Artificial Intelligence* (pp. 248-253).
- [7] Mishra, A. K., Siddalingaswamy, P. C. (2017, April). Analysis of tree based search techniques for solving 8-puzzle problem. In *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)* (pp. 1-5). IEEE.
- [8] Nayak, D. (2014). Analysis and Implementation of Admissible Heuristics in 8 Puzzle Problem (Doctoral dissertation).
- [9] Igwe, K., Pillay, N., Rae, C. (2013, October). Solving the 8-puzzle problem using genetic programming. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference* (pp. 64-67).