

# Mid Assignment Report

CSE-0408 Summer 2021

Ahasanul Bari Rifat

Department of Computer Science and Engineering

State University of Bangladesh (SUB)

Dhaka, Bangladesh

ahsanulbari.rifat88@gmail.com

**Abstract—Code 1 :** In practice, an incomplete heuristic search nearly always finds better solutions if it is allowed to search deeper, i.e. expand and heuristically evaluate more nodes in the search tree and determine the best path to take next. The heuristic function is a way to inform the search about the direction to a goal. It provides an informed way to guess which neighbor of a node will lead to a goal. There is nothing magical about a heuristic function. It must use only information that can be readily obtained about a node.

**Code 2 :** As discussed earlier, Breadth-First Search (BFS) is an algorithm used for traversing graphs or trees. Traversing means visiting each node of the graph. Breadth-First Search is a recursive algorithm to search all the vertices of a graph or a tree. BFS in python can be implemented by using data structures like a dictionary and lists. Breadth-First Search in tree and graph is almost the same. The only difference is that the graph may contain cycles, so we may traverse to the same node again. Index Terms—heuristic, puzzle traverse, cycle, graph

*Index Terms*—C++ , python

## I. INTRODUCTION

**Code 1 :** Breadth First Traversal (or Search) for a graph is similar to Breadth First Traversal of a tree (See method 2 of this post). The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array. For simplicity, it is assumed that all vertices are reachable from the starting vertex. For example, in the following graph, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don't mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Breadth First Traversal of the following graph is 2, 0, 3, 1.

**Code 2 :** Many problems, such as game-playing and path-finding, can be solved by search algorithms. To do so, the problems are represented by a search graph or tree in which the nodes correspond to the states of the problem. In this assignment we are going to implement a algorithms to solve 8 puzzle problem.

## II. LITERATURE REVIEW

8 puzzle solve

. In 2012 S. et. al. [4] proposed new resolution for solving N-queens by using combination of DFS (Depth First Search) and BFS (Breadth First Search) techniques. The proposed

algorithm act based on placing queens on chess board directly. The results report the performance and run time of this approach. Breadth-First Search The two variants of Best First Search are Greedy Best First Search and A\* Best First Search. Greedy BFS: Algorithm selects the path which appears to be the best, it can be known as the combination of depth-first search and breadthfirst search. Greedy BFS makes use of Heuristic function.

## III. PROPOSED METHODOLOGY

Here i Discuss BFS Algorithm:

1. for each  $u$  in  $V$  s
2. do  $color[u] \leftarrow WHITE$
3.  $d[u] \leftarrow infinity$
4.  $[u] \leftarrow NIL$
5.  $color[s] \leftarrow GRAY$
6.  $d[s] \leftarrow 0$
7.  $[s] \leftarrow NIL$
8.  $Q \leftarrow$
9.  $ENQUEUE(Q, s)$
- 10 while  $Q$  is non-empty.

## 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.

[illegible][illegible]

The screenshot shows a Jupyter Notebook interface with a single cell containing Python code. The code defines a neural network with two layers, each with 10 nodes. It includes functions for the sigmoid activation function, the forward pass, and the backward pass (backpropagation). The code is as follows:

```

import numpy as np
import sys

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def forward_pass(weights, biases, inputs):
    # Layer 1
    z1 = np.dot(weights[0], inputs) + biases[0]
    a1 = sigmoid(z1)
    # Layer 2
    z2 = np.dot(weights[1], a1) + biases[1]
    a2 = sigmoid(z2)
    return a2

def backward_pass(weights, biases, inputs, target, a1):
    # Layer 2 error
    e2 = a2 - target
    # Layer 1 error
    e1 = a1 * (1 - a1) * (weights[1] * e2)
    # Gradients
    dw1 = e1 * a1 * (1 - a1) * inputs
    dw2 = e2 * a2 * (1 - a2) * a1
    db1 = e1 * a1 * (1 - a1)
    db2 = e2 * a2 * (1 - a2)
    return dw1, dw2, db1, db2

# Initialize weights and biases
weights = np.random.randn(2, 10)
biases = np.random.randn(2, 1)

# Inputs and target
inputs = np.random.randn(1, 10)
target = np.random.randn(1, 1)

# Forward pass
a2 = forward_pass(weights, biases, inputs)

# Backward pass
dw1, dw2, db1, db2 = backward_pass(weights, biases, inputs, target, a1)

# Print results
print("Forward pass output: {}".format(a2))
print("Backward pass gradients: dw1={}, dw2={}, db1={}, db2={}".format(dw1, dw2, db1, db2))

```

The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Windows, Help), a toolbar with icons for file operations and execution, and a status bar at the bottom showing the system clock and temperature.

The screenshot displays a Jupyter Notebook environment with the following components:

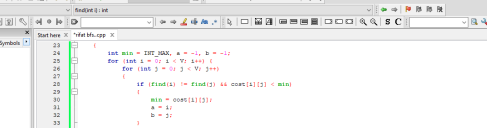
- Top Bar:** Shows the file explorer with a list of files including `01.py`, `02.py`, `03.py`, `04.py`, `05.py`, `06.py`, `07.py`, `08.py`, `09.py`, `10.py`, `11.py`, `12.py`, `13.py`, `14.py`, `15.py`, `16.py`, `17.py`, `18.py`, `19.py`, `20.py`, `21.py`, `22.py`, `23.py`, `24.py`, `25.py`, `26.py`, `27.py`, `28.py`, `29.py`, `30.py`, `31.py`, `32.py`, `33.py`, `34.py`, `35.py`, `36.py`, `37.py`, `38.py`, `39.py`, `40.py`, `41.py`, `42.py`, `43.py`, `44.py`, `45.py`, `46.py`, `47.py`, `48.py`, `49.py`, `50.py`, `51.py`, `52.py`, `53.py`, `54.py`, `55.py`, `56.py`, `57.py`, `58.py`, `59.py`, `60.py`, `61.py`, `62.py`, `63.py`, `64.py`, `65.py`, `66.py`, `67.py`, `68.py`, `69.py`, `70.py`, `71.py`, `72.py`, `73.py`, `74.py`, `75.py`, `76.py`, `77.py`, `78.py`, `79.py`, `80.py`, `81.py`, `82.py`, `83.py`, `84.py`, `85.py`, `86.py`, `87.py`, `88.py`, `89.py`, `90.py`, `91.py`, `92.py`, `93.py`, `94.py`, `95.py`, `96.py`, `97.py`, `98.py`, `99.py`, `100.py`, `101.py`, `102.py`, `103.py`, `104.py`, `105.py`, `106.py`, `107.py`, `108.py`, `109.py`, `110.py`, `111.py`, `112.py`, `113.py`, `114.py`, `115.py`, `116.py`, `117.py`, `118.py`, `119.py`, `120.py`, `121.py`, `122.py`, `123.py`, `124.py`, `125.py`, `126.py`, `127.py`, `128.py`, `129.py`, `130.py`, `131.py`, `132.py`, `133.py`, `134.py`, `135.py`, `136.py`, `137.py`, `138.py`, `139.py`, `140.py`, `141.py`, `142.py`, `143.py`, `144.py`, `145.py`, `146.py`, `147.py`, `148.py`, `149.py`, `150.py`, `151.py`, `152.py`, `153.py`, `154.py`, `155.py`, `156.py`, `157.py`, `158.py`, `159.py`, `160.py`, `161.py`, `162.py`, `163.py`, `164.py`, `165.py`, `166.py`, `167.py`, `168.py`, `169.py`, `170.py`, `171.py`, `172.py`, `173.py`, `174.py`, `175.py`, `176.py`, `177.py`, `178.py`, `179.py`, `180.py`, `181.py`, `182.py`, `183.py`, `184.py`, `185.py`, `186.py`, `187.py`, `188.py`, `189.py`, `190.py`, `191.py`, `192.py`, `193.py`, `194.py`, `195.py`, `196.py`, `197.py`, `198.py`, `199.py`, `200.py`, `201.py`, `202.py`, `203.py`, `204.py`, `205.py`, `206.py`, `207.py`, `208.py`, `209.py`, `210.py`, `211.py`, `212.py`, `213.py`, `214.py`, `215.py`, `216.py`, `217.py`, `218.py`, `219.py`, `220.py`, `221.py`, `222.py`, `223.py`, `224.py`, `225.py`, `226.py`, `227.py`, `228.py`, `229.py`, `230.py`, `231.py`, `232.py`, `233.py`, `234.py`, `235.py`, `236.py`, `237.py`, `238.py`, `239.py`, `240.py`, `241.py`, `242.py`, `243.py`, `244.py`, `245.py`, `246.py`, `247.py`, `248.py`, `249.py`, `250.py`, `251.py`, `252.py`, `253.py`, `254.py`, `255.py`, `256.py`, `257.py`, `258.py`, `259.py`, `260.py`, `261.py`, `262.py`, `263.py`, `264.py`, `265.py`, `266.py`, `267.py`, `268.py`, `269.py`, `270.py`, `271.py`, `272.py`, `273.py`, `274.py`, `275.py`, `276.py`, `277.py`, `278.py`, `279.py`, `280.py`, `281.py`, `282.py`, `283.py`, `284.py`, `285.py`, `286.py`, `287.py`, `288.py`, `289.py`, `290.py`, `291.py`, `292.py`, `293.py`, `294.py`, `295.py`, `296.py`, `297.py`, `298.py`, `299.py`, `300.py`, `301.py`, `302.py`, `303.py`, `304.py`, `305.py`, `306.py`, `307.py`, `308.py`, `309.py`, `310.py`, `311.py`, `312.py`, `313.py`, `314.py`, `315.py`, `316.py`, `317.py`, `318.py`, `319.py`, `320.py`, `321.py`, `322.py`, `323.py`, `324.py`, `325.py`, `326.py`, `327.py`, `328.py`, `329.py`, `330.py`, `331.py`, `332.py`, `333.py`, `334.py`, `335.py`, `336.py`, `337.py`, `338.py`, `339.py`, `340.py`, `341.py`, `342.py`, `343.py`, `344.py`, `345.py`, `346.py`, `347.py`, `348.py`, `349.py`, `350.py`, `351.py`, `352.py`, `353.py`, `354.py`, `355.py`, `356.py`, `357.py`, `358.py`, `359.py`, `360.py`, `361.py`, `362.py`, `363.py`, `364.py`, `365.py`, `366.py`, `367.py`, `368.py`, `369.py`, `370.py`, `371.py`, `372.py`, `373.py`, `374.py`, `375.py`, `376.py`, `377.py`, `378.py`, `379.py`, `380.py`, `381.py`, `382.py`, `383.py`, `384.py`, `385.py`, `386.py`, `387.py`, `388.py`, `389.py`, `390.py`, `391.py`, `392.py`, `393.py`, `394.py`, `395.py`, `396.py`, `397.py`, `398.py`, `399.py`, `400.py`, `401.py`, `402.py`, `403.py`, `404.py`, `405.py`, `406.py`, `407.py`, `408.py`, `409.py</`

The screenshot shows the Visual Studio IDE with the following components:

- Top Menu Bar:** File, Edit, View, Search, Project, Build, Debug, Format, Extensions, Tools, Team, Plugins, Daylight, Settings, Help.
- Toolbar:** Standard Visual Studio toolbar with icons for file operations, editing, and running.
- Project Explorer:** Shows the file structure with 'Viterbi\_Mat.cpp' selected.
- Solution Explorer:** Shows the 'Viterbi\_Mat.cpp' file.
- Code Editor:** Displays the C++ code for 'Viterbi\_Mat.cpp'. The code includes a main function that reads input, initializes a Viterbi object, and prints the results. The code is as follows:
 

```

1 // Viterbi_Mat.cpp
2 #include <iostream>
3 using namespace std;
4 int V;
5 int parent[1000];
6 int state[1000][1000];
7 int find(int i)
8 {
9     while (parent[i] != i)
10         i = parent[i];
11     return i;
12 }
13 void init(int i, int j)
14 {
15     int a = find(i);
16     int b = find(j);
17     parent[i] = b;
18 }
19 void PRINT()
20 {
21     int nstates = V;
22     int nseq_count = 0;
23     while (seq_count < V - 1)
      
```
- Output Window:** Shows the output of the program, which is the sequence of states and the count of sequences.
- Bottom Bar:** Shows the status bar with 'C/C++', 'Windows (x64)', 'WINDOWS 10', 'Line 6, Col 1, Pst 1', and 'SYSTEM'.



The screenshot displays the Visual Studio Code editor with a C++ file named 'yafefMf.cpp'. The code is as follows:

```

1 // C++ program to find maximum element in an array
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int n;
8     cin >> n;
9     int arr[n];
10    for (int i = 0; i < n; i++)
11        cin >> arr[i];
12
13    int maxi = arr[0];
14    for (int i = 1; i < n; i++)
15        if (arr[i] > maxi)
16            maxi = arr[i];
17
18    cout << "Maximum element is: " << maxi << endl;
19    return 0;
20 }

```

The 'Run and Debug' window at the bottom shows the program's execution. The 'Cygwin' terminal displays the input '5' and the output 'Maximum element is: 5'. The 'Output' window shows the same output. The 'Run and Debug' window also includes a 'Variables' tab showing the state of the program during execution.

The screenshot shows the Visual Studio Code IDE with the following details:

- Top Bar:** Displays the file name 'main.cpp - CodeBlocks2018', a search bar, and various icons for file operations (Save, Open, Close, etc.).
- Left Panel:** Contains the 'Explorer' view showing the project structure with 'main.cpp' selected.
- Main Editor:** Displays the source code of 'main.cpp'. The code defines a recursive function 'fibonacci' that calculates the nth Fibonacci number. The function uses a base case of 1 for n <= 1 and a recursive call for n > 1. The main function calls 'fibonacci(10)' and prints the result.
- Output Console:** Located at the bottom, it shows the output of the program: '10th fibonacci number is: 55'.
- Taskbar:** At the very bottom, it shows the Windows taskbar with the taskbar icon, system clock (10:42 AM, 10/2/2018), and system tray icons (network, volume, battery).

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

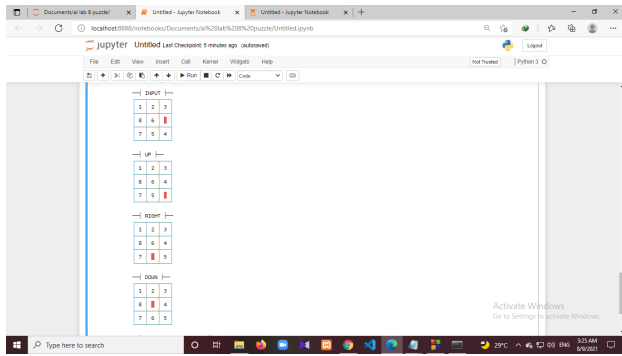


Fig. 8.

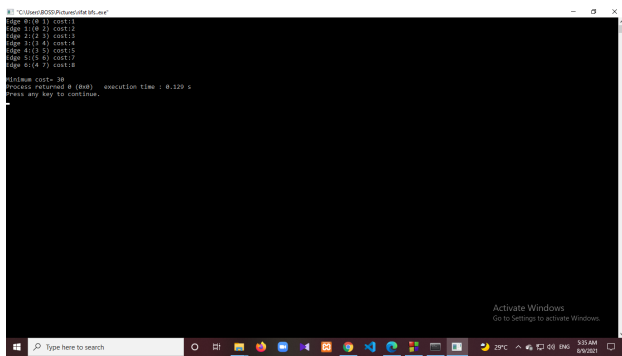


Fig. 9.

## IX. CONCLUSION

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] Arbolea, 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] Reinefeld, A. (1993, August). Complete Solution of the Eight-Puzzle and the Benefit of Node Ordering in IDA\*. In *International Joint Conference on Artificial Intelligence* (pp. 248-253).
- [4] 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).