UNIVERSITI TUNKU ABDUL RAHMAN

ACADEMIC YEAR 2021/2022

MAY 2021 TRIMESTER

ASSIGNMENT

<u>ANSWER SCRIPT</u>

Candidate is required to fill in ALL the information below:

Group Leader Name : (as stated in Student Identity Card)	Ng Khee Long		
Faculty /Institute/ Centre:	LKC FES	Programme :	SOFTWARE ENGINEERING
Group Leader ID:	1802451		
Course Code :	UECS2153	Course Description :	Artificial Intelligence
Group Member Name:	Goh Pei Jin	Group Member ID:	1802410
Group Member Name:	Gervin Fung Da Xuen	Group Member ID:	1801655

DECLARATION STATEMENT

We, Ng Khee Long, Goh Pei Jin, Gervin Fung Da Xuen,
with Student ID Nos.1802451, 1802410, 1801655 hereby solemnly and sincerely declare and confirm that we have read, understood and shall abide and comply with all laws, rules, regulations, guidelines and lawful instruction of the University and its staff in relation to the commencement of any assessment / examination during our programme of study in Universiti Tunku Abdul Rahman.

We hereby declare that our submission for all assessment / examination during our programme of study in the University shall be based on our original work, not plagiarised from any source(s) except for citations and quotations which have been duly acknowledged. We are fully aware that students who are suspected of violating this pledge are liable to be referred to the Examination Disciplinary Committee of the University.

We further certify that we have read and understand the above guidelines and agree to abide by the above declarations.

Long　　　　[illegible]　　　　[illegible]

Names: Ng Khee Long　　　　Goh Pei Jin　　　　Gervin Fung Da Xuen

Q1

Given that the Student ID of this group's leader - Ng Khee Long is 1802451.

```
Student ID : 1802451. Number from Modulo 8 + 1:  4
```

Figure 1.1 Resulting Number

Referring to Figure 1.1 above we obtained number of 4, hence the historical figure we would discuss about is ***Frank Rosenblatt***(4)

(a) Summarize (maximum 1/3 of an A4 page) your assigned figure's life history (non-AI related). (10 marks)

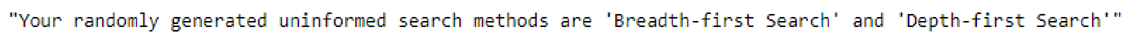
Frank Rosenblatt was born on July 11, 1928, in New Rochelle of New York and graduated from both Bronx High School of Science in 1946 and Cornell University in 1950 and 1956, where he attained both his Artium Baccalaureus (A.B.) and Doctor of Philosophy (PhD), Frank Rosenblatt was a genius with broad interests, as he travelled to Cornell's Ithaca campus in 1959 and had successfully become both the director of the Cognitive Systems Research Program and a lecturer in the Psychology Department (Emlen, Howland, O'Brien, n.d.). Fascinated with the transfer of learned behaviour, Frank Rosenblatt decided to inject the brain extracts from trained to naive rats, which would later be published substantially in later years. He became field representative for the Graduate Field of Neurobiology and Behavior in 1970, and a year later, he shared the acting chairmanship of the Section of Neurobiology and Behavior. On his 43rd birthday, a boating accident in the Chesapeake Bay resulted in Frank Rosenblatt's death (Emlen, Howland, O'Brien, n.d.).

(b) Explain the figure's primary contribution(s) to AI, with special focus on commercial or real-life technologies which derive from those contributions. The answer to Q1(b) should take at max 2/3 of an A4 page. (20 marks)

Frank Rosenblatt began investigating perceptrons, a kind of artificial neural network, in 1957, using the McCulloch-Pitts neuron and Hebb's discoveries (Rosenblatt, 1961). Perceptron is a single-layer neural network inspired by the way neurons interact in the brain (Lefkowitz, 2019). It is an algorithm that is applied in supervised learning as one of the binary classifiers, therefore it allows neurons to have the capabilities to learn and processes in the training set one per time (Simplilearn, 2021). The perceptron was only able to work with linear separation of data points unless we have activation functions, so it is only able to classify within two classes. It consists of a single neuron with adjustable synaptic weights and bias. The formula can be defined as $Z = \sum_{i=1}^{m} W_i X_i + b$, **w** = vector of real-valued weights, **b** = bias, **x** = vector of input x values and **m** = number of inputs to the Perceptron. Z is the total of these three inputs, represented as 0,1 or -1,1 depending on the activation function, where 1 indicates taken and 0 or -1 indicates not taken. The perceptron can auto-deduce the appropriate weights from training data, by tweaking itself to produce a better informed prediction, and it will get more when iterated hundreds or millions of times (Lefkowitz, 2019). Rosenblatt also used the phrase "back propagation error correction" to describe two-layer networks which could be applied to multilayer networks, which were introduced by Belmont Farley and Wesley Clark. This phrase “back-propagation” has been presented and improved until now which enables the neural network to train the layer to improve accuracy by going backward through layers (Copeland, 2015). Rosenblatt's perceptron approach failed to achieve his goal of allowing the machine to recognize the objects. However, after 60 years, the researchers found that applying the multiple layers in the perceptron approach while training the networks, has allowed the object recognition idea to be achieved. His discoveries have provided insight into the world, and the algorithm he developed is still essential to how we train deep neural networks today (Lefkowitz, 2019). Deep learning has been achieved in many areas by adopting perceptrons as a fundamental component, such as image recognition, facial recognition, voice recognition, etc. These have enabled us to detect objects in images and can search objects by visual searching. It also allowed us to recognise faces by extracting facial landmarks and recognise speech by analysing and pattern recognition. All these have implemented the idea of perceptron by applying multiple times to build a neural network to train the model and to be used as a tool to assist the application and web.

<u>Q2</u>

(a) Describe how your assigned uninformed search methods work conceptually



```
"Your randomly generated uninformed search methods are 'Breadth-first Search' and 'Depth-first Search'"
```

Figure 2.1

As shown by the figure above, the uninformed search methods assigned are Breadth-First-Search (*BFS*) and Depth-First-Search (*DFS*). The next paragraph will discuss some of the common terms used in describing the conceptual work of both searching methods.

1. **Algorithm** - *A set of rules written by Programmers/Developers to take one or more inputs and then performs inner data manipulations and returns one or more outputs, which may or may not be in the form of data structure.*
2. **Tree** - *A non-linear data structure with multiple nodes. In other words, it's a data structure in which elements are arranged in multiple levels.*
3. **Node** - *Part of a* ***Tree*** *from which one or more* ***Nodes*** *emerge.*

The next subsection will discuss the conceptual work of BFS and DFS in detail with given examples.

1. **BFS**

BFS is an algorithm that is used for traversing a tree to search for a node that satisfies given criteria. BFS explores all the available nodes at a given depth before any nodes at the next level are explored (See figure below).

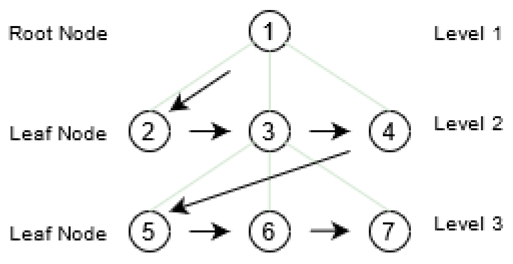


Figure 2.1.1 BFS conceptual work

As shown by the figure, the light-grey colour branch is the tree branch of the BFS algorithm. The black arrow is the traversal direction of the BFS algorithm. Node 1 is the root node of the BFS algorithm, and it can discover/explore can be either one of the leaf nodes available at *Level 2*. It would be simpler to go with the number given. Hence let's assume that the first leaf node it explores is node 2. The question to ask for BFS right now is, "*Are there any other nodes that branch from the parent node of the current node?*", in this case, both nodes 3 and 4 branch from the same parent node of node 2. Hence we will be moving to node 3 and repeat the same question until all the nodes at the same level are searched.

When we reach the last branch node of node 1 at *Level 2*, we check if there is another level for the tree. If there is, we will now be moving to another level in the tree, which starts on node 5 at *Level 3* because we have already *explored all the nodes at a given depth*. Notice that the question to ask when each node is explored is the same. Hence, following the pattern that we can observe, the transversal order of BFS is 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7.

2. DFS

Similar to BFS, DFS is an algorithm used for traversing a tree to search for a node that satisfies given criteria. However, the difference is that DFS expands the deepest node in the current frontier of the given search tree, refer to the figure shown below.

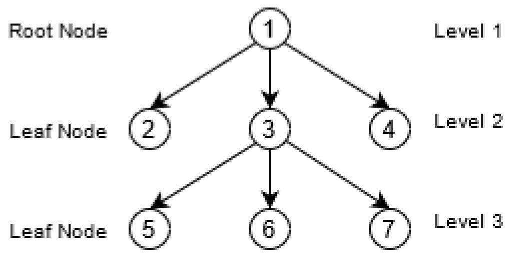


Figure 2.2.1 DFS conceptual work

Referring to the figure above, node 1 is the root node or the starting node. The first node it can discover/explore can be either one of the leaf nodes available at *Level 2*. It would be simpler to go with the number given. Hence let's assume that the first leaf node it explores is node 2. So far, all the steps are similar to that of BFS. However, this is where both searching algorithms differ. The question to ask for DFS is, "*Are there any other descending nodes that branches from current node*", we can see that there's no other leaf node that branches from node 2. Hence, we move back to the root node.

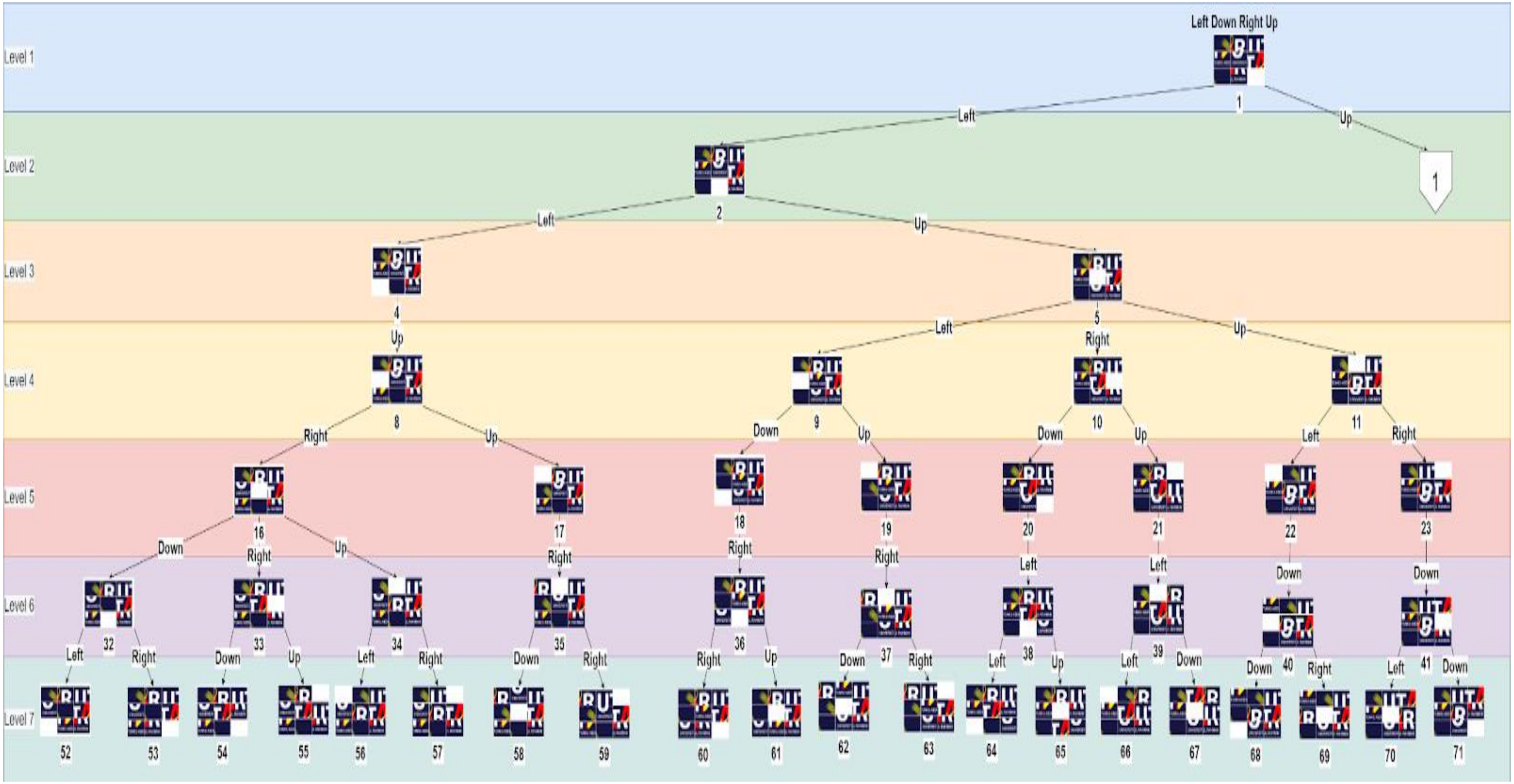
Now we move to node 3; repeating the same question, we found that 3 nodes are branches from node 3. Similar to the root node, we can discover/explore either one of the leaf nodes available at *Level 3*. Let's assume again that the first node it explores is node 5, and we repeat the same question, there's no leaf node as node 5 has no descendant nodes. Hence it moves back to node 3 to see if there are any deeper level nodes, which are nodes 6 and 7. It's worth noticing that the same scenario repeats again and again.

To summarise, DFS will always explore the deepest node available. Hence, the traversal order of DFS is 1 -> 2 -> 3 -> 5 -> 6 -> 7 -> 4.

(b)

The following DFS and BFS search trees are drawn with 91 nodes for each search method. The search tree consists of a depth with 7 levels. Please note that no solution paths are found for the search trees given, and it is not a complete search tree for each of the search methods. Both will be represented by the same tree with different number labelling. This search will select based on left, down, right and up. The repeated puzzle will not be considered. To see clearer pictures you can go to https://drive.google.com/drive/folders/1B6MaxAOSjbh06iDp0GG6GOaYBkBU7a7T?usp=sharing to download the png files.

1. **BFS**


Level 1
Level 2
Level 3
Level 4
Level 5
Level 6
Level 7
Left Down Right Up
1
Left
Up
1
2
Left
Up
4
5
Up
Left
Right
Up
8
9
10
11
Right
Up
Down
Up
Down
Up
Left
Right
16
17
18
19
20
21
22
23
Down
Right
Up
Right
Right
Right
Left
Left
Down
Down
32
33
34
35
36
37
38
39
40
41
Left
Right
Down
Up
Left
Right
Down
Right
Right
Up
Down
Right
Left
Up
Left
Down
Down
Right
Left
Down
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

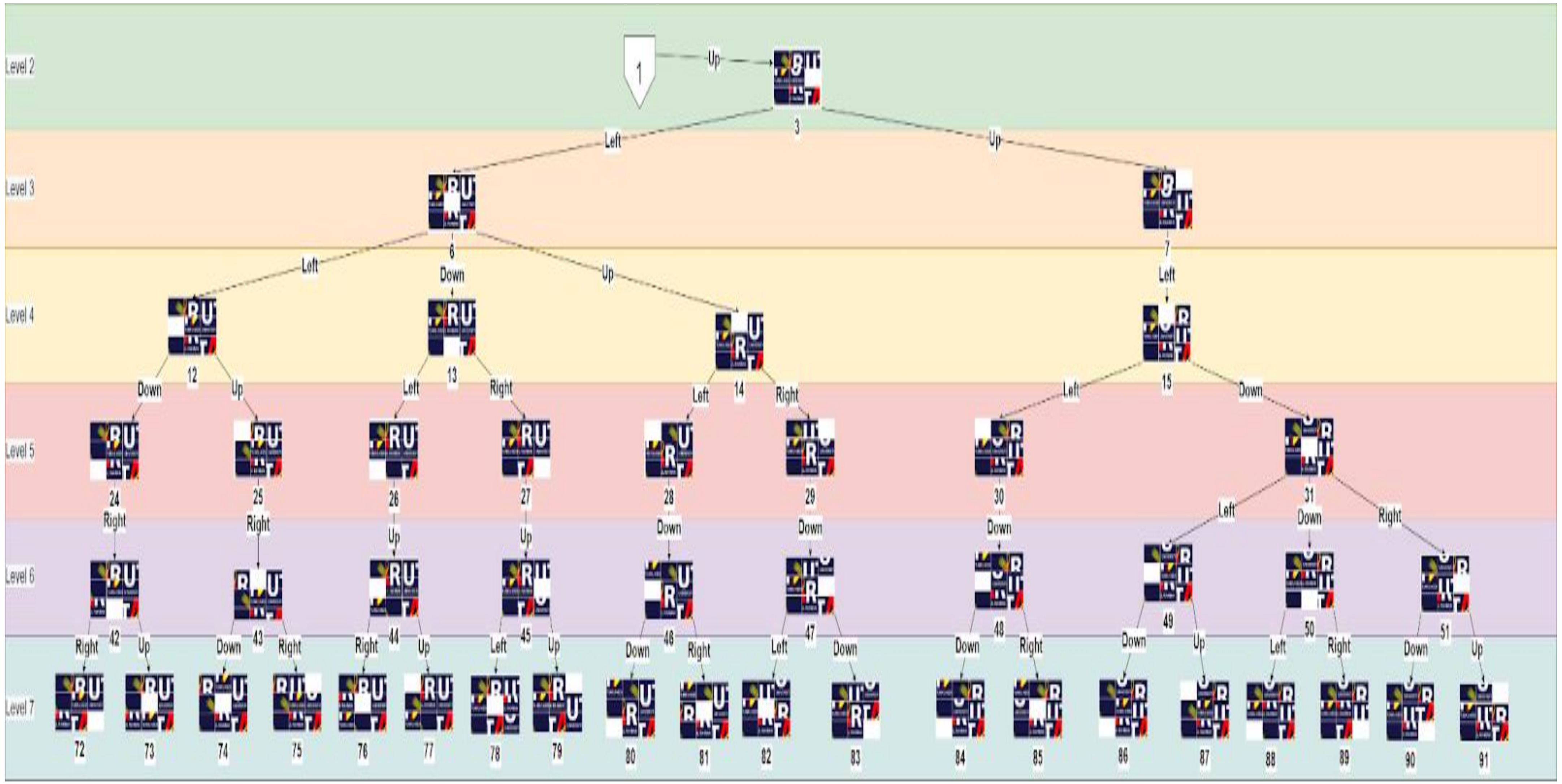
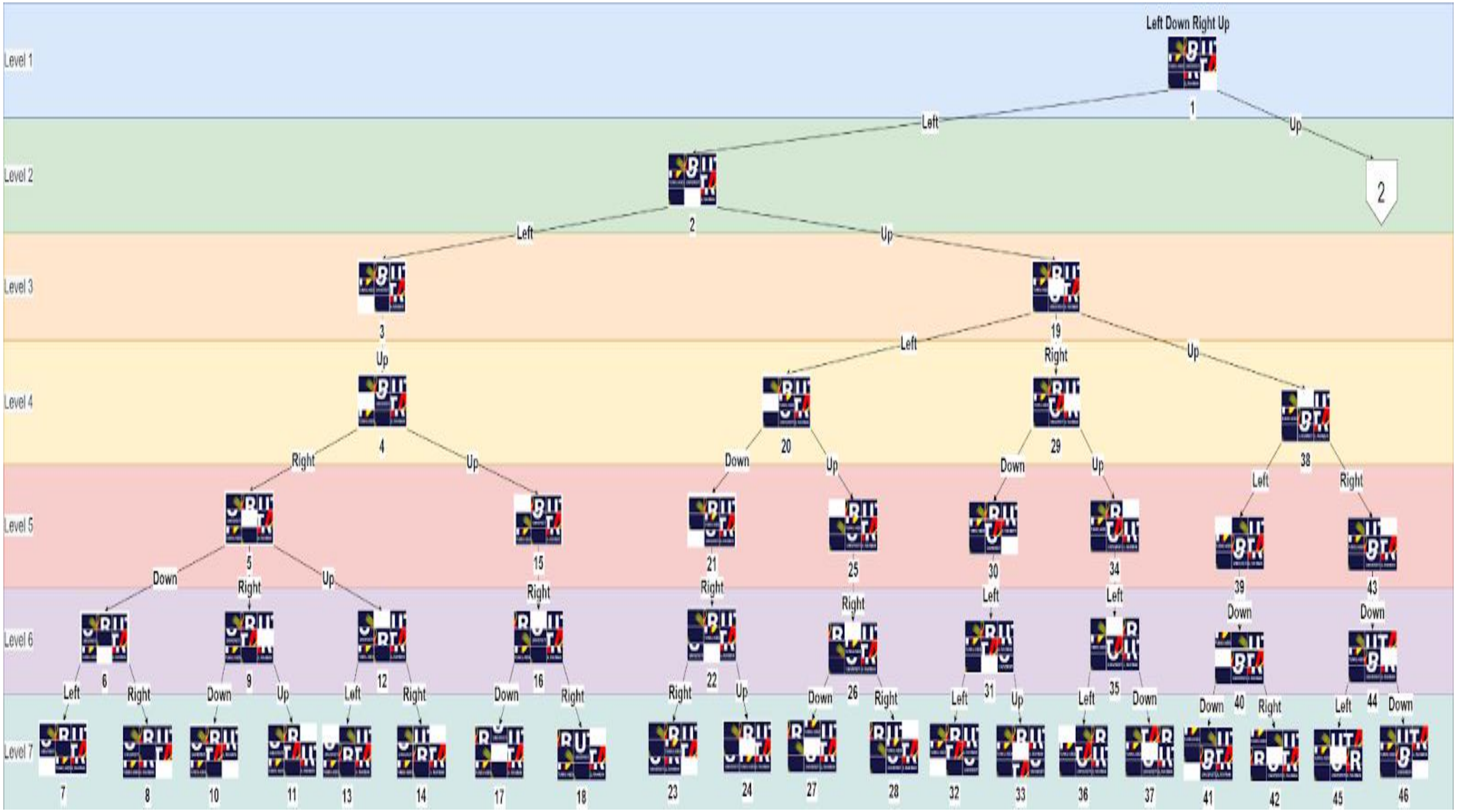


Figure 3.1 BFS

Figure 3.1 shows the nodes for the BFS. It will traverse node level by level starting from level 1, followed by level 2 until level 7. The numbering is labeled below the image to show how the node travels, which is 1 to 91.

2. DFS


Level 1
Level 2
Level 3
Level 4
Level 5
Level 6
Level 7
Left Down Right Up
1
Left
Up
2
2
Left
Up
3
Up
4
Right
Up
5
Down
Right
Up
6
Left
Right
7
8
9
Down
Up
10
11
12
Left
Right
13
14
15
Right
16
Down
Right
17
18
19
Left
Right
Up
20
Down
Up
21
Right
22
Right
Up
23
24
25
Right
26
Down
Right
27
28
29
Down
Up
30
Left
31
Left
Up
32
33
34
Left
35
Left
Down
36
37
38
Left
Right
39
Down
40
Down
Right
41
42
43
Down
44
Left
Down
45
46

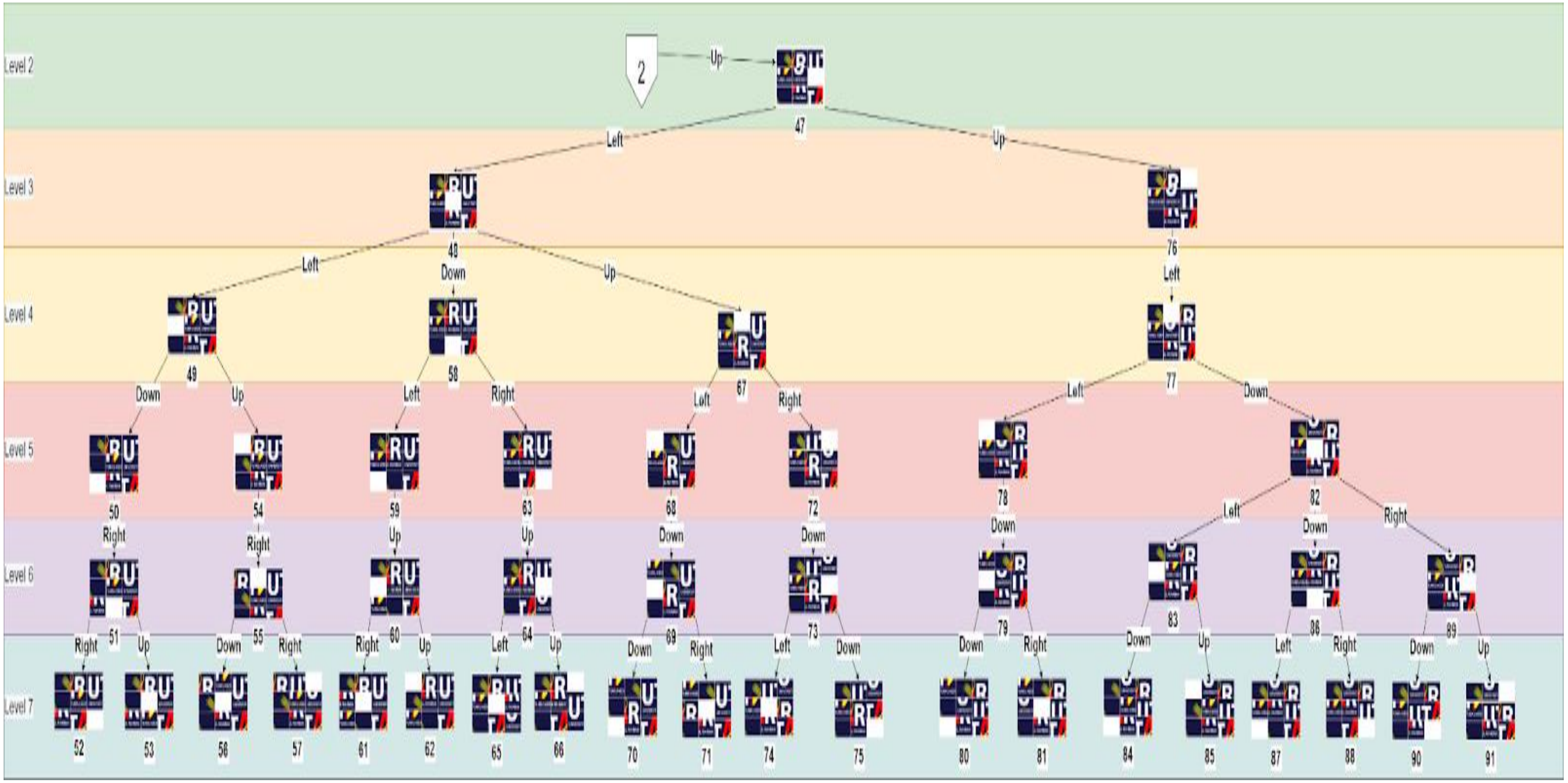
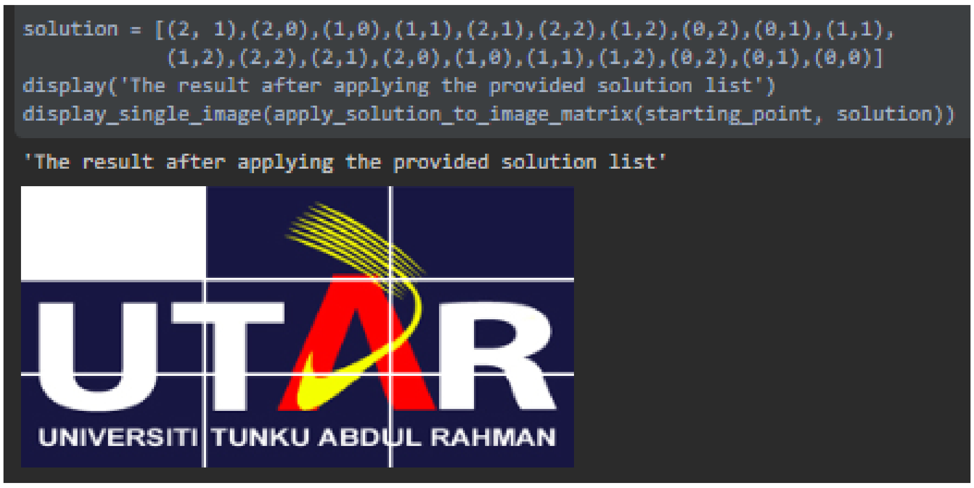


Figure 3.2 DFS

Figure 3.2 shows the nodes for the DFS. It will traverse nodes from the leftmost path until there is no node, then jump to the right path. The numbering is labelled below the image to show how the node travels.

(c) Solutions of the puzzle

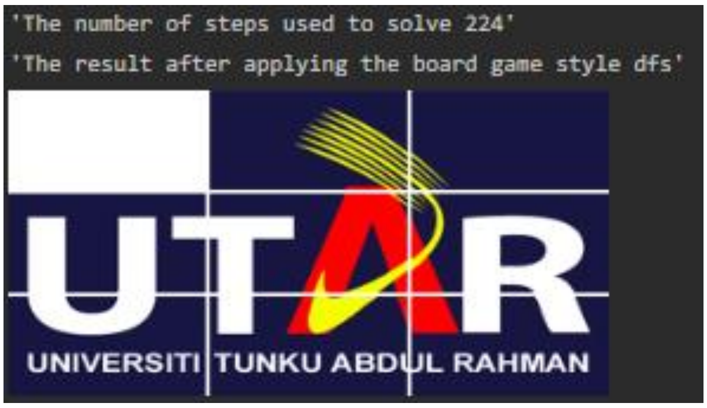
After trying a few search methods, we applied the A* search method and is able to generate the answer with 20 moves. The results for A* search, DFS, conventional DFS search and BFS search is at shown below.



```
solution = [(2, 1),(2,0),(1,0),(1,1),(2,1),(2,2),(1,2),(0,2),(0,1),(1,1),
            (1,2),(2,2),(2,1),(2,0),(1,0),(1,1),(1,2),(0,2),(0,1),(0,0)]
display('The result after applying the provided solution list')
display_single_image(apply_solution_to_image_matrix(starting_point, solution))
```

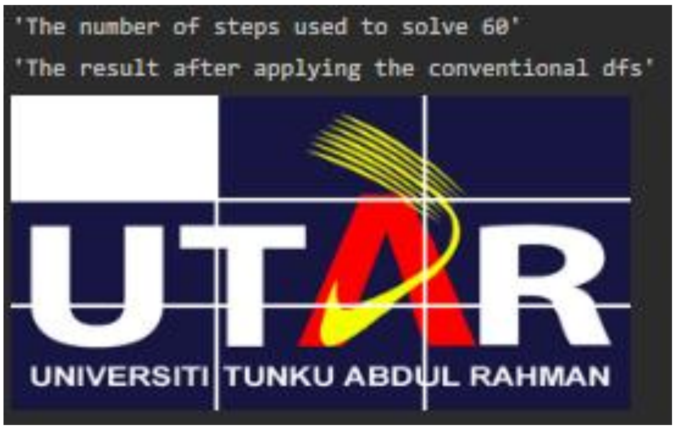
```
'The result after applying the provided solution list'
```

Figure 3.3 A* search result and moves



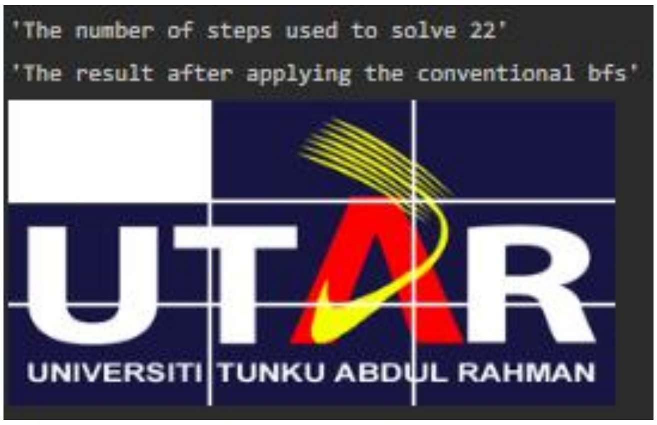
```
'The number of steps used to solve 224'
'The result after applying the board game style dfs'
```

Figure 3.4 DFS search result and moves



```
'The number of steps used to solve 60'
'The result after applying the conventional dfs'
```

Figure 3.5 Conventional DFS search result and moves



```
'The number of steps used to solve 22'
'The result after applying the conventional bfs'
```

Figure 3.6 BFS search result and moves

The detailed list of moves for two DFS search BFS search can be seen in the code given in the cell 4 in jupyter notebook.

(d) Evaluate the overall performance (efficiency, efficacy, and any other criteria you consider important) of both your assigned search methods in comparison to each other.

There are a total of 5 criteria that are deemed important to compare both searching algorithms, which are *efficiency (divided into time and space complexity), memory consumption, efficacy, and completeness.*

1. Time Complexity

Time complexity is commonly used as the golden standard when it comes to comparing algorithms, because an efficient algorithm will use the least amount of time to achieve its objectives, of course, there is also space complexity that should be taken into consideration which will be discussed in the next subsection, however, we always consider the worst time complexity which is the longest time used to solve the problem scenario for comparison purpose. Hence, time complexity can be defined as the total amount of time required for an algorithm to complete its execution.

The time complexity of BFS and DFS search in the worst-case scenario is O (b^d), given that b is the branching factor of a search tree and d is the maximum depth. Hence in terms of time complexity, both are equally good, but that’s not the whole story. It’s important to note that having the same time complexity does not mean the amount of time required to run the program will be the same, as the resources of a PC may vary from time to time. However, time complexity gives us an idea of the amount of time required to complete the algorithm.

2. Space Complexity and Memory Consumption

Space complexity is another standard used when comparing the space efficiency of an algorithm because an efficient algorithm will use the least amount of space to achieve its objectives. Hence space complexity is defined as the amount of spaces required by an algorithm to complete its execution.

Given that B is the branching factor of a search tree and M is the maximum length of path, then the space complexity of BFS and DFS in the worst-case scenario will be O (B^M) and O (BM) respectively, hence in this case, DFS triumph over BFS in terms of space complexity.

This can lead us to conclude that the memory consumption of DFS is less than that of BFS, because DFS implemented a **Last-In-First-Out** (LIFO) data structure and explore the deepest node in the current frontier of the given search tree until it cannot proceed anymore and backtracks from there. Hence DFS only has to remember a single path with all the unexplored nodes. Whereas BFS implemented a **First-In-First-Out** (FIFO) data structure because it explores all the available nodes at a given depth of the parent node and keeps them in memory. Hence, in terms of memory consumption, DFS is more memory efficient than BFS.

3. Efficacy

Efficacy can be defined as the optimality of an algorithm, and it’s worth noting that optimality does not mean the algorithm used is fast. It simply means it can produce the most optimal path that will lead to a solution because the fastest returned solution may not be the best solution and vice versa. In other words, an optimal algorithm will return the best solution if it manages to find a solution. DFS is therefore not optimal due to the fact that the number of vertices it took to reach the solution is too high, or the cost to search for a solution is too high. BFS, on the other hand, is optimal if the edges are unweighted. Hence, in terms of efficacy, BFS towers over DFS.

4. Completeness

Completeness of an algorithm can be defined as an algorithm that is guaranteed to be able to search for a solution in a finite amount of time, if and only if at least one solution exists. BFS is considered as a complete algorithm because it will explore all the nodes at a given depth before moving on to the next depth. As such, if a solution exists at a given depth or height, then BFS will eventually be able to search for the solution at the node of the given depth or height. However, DFS is considered as an incomplete algorithm because it’s only complete if the search tree is finite, if the search tree is infinite, it will keep on exploring its deepest node. Hence in terms of completeness, BFS is better than DFS.

Conclusion

In conclusion, BFS is preferred if the branching factor is small as the space complexity will be too high otherwise. On the other hand, DFS is preferred if the solution is known to be at a certain depth, thus the DFS algorithm would be modified to search only until a certain depth.

References

Copeland, B.J., 2015. *Perceptrons.* [online] Available at: <https://www.britannica.com/technology/perceptrons> [Accessed 16 July 2021]

Emlen, S.T., Howland, H.C., O’Brien, R.D., n.d. *Frank Rosenblatt.* [online] Available at: <https://ecommons.cornell.edu/bitstream/handle/1813/18965/Rosenblatt_Frank_1971.pdf;jsessionid=4E1F30C6D24133367BF37BD1121E4107?sequence=2> [Accessed 16 July 2021]

Lefkowitz, M., 2019. *Professor’s perceptron paved the way for AI – 60 years too soon*. [online] <https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon>[Accessed 17 July 2021].

Rosenblatt, F., 1961. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Cornell Aeronautical Lab Inc Buffalo NY.

Simplilearn, 2021. *What is Perceptron: A Beginners Guide for Perceptron*. [online] Available at:

<https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron> [Accessed 17 July 2021]