CURRENT RESULT

TIME: 01:24:02
MOVES: 116

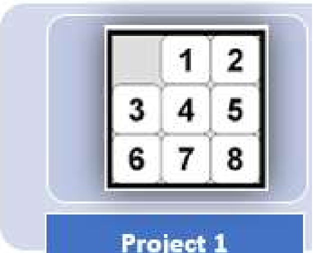
PERSONAL BEST

TIME: 01:07:04
MOVES: 110

Implementation and software

Project#1 (5 points)

Project 1



	1	2
3	4	5
6	7	8

Project 1
8-puzzle problem

Step 1 Define actions set & Generating tree

Step 2 Implement simple Search algorithm*

Step 3

Step 5

Dates MM/DD

Deliverables

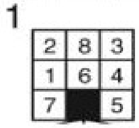
- Source code
- Output file

*Optional

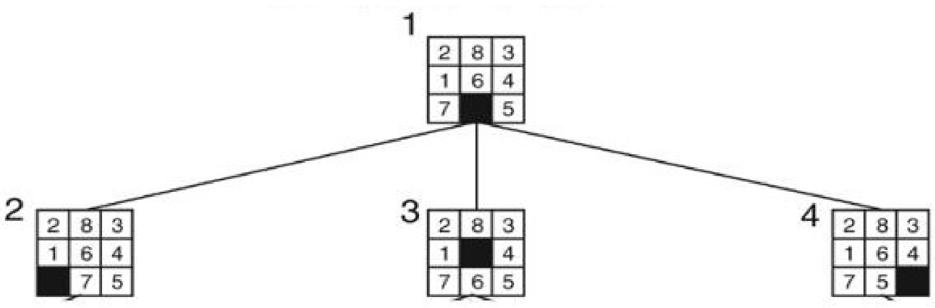
Project -1 Description

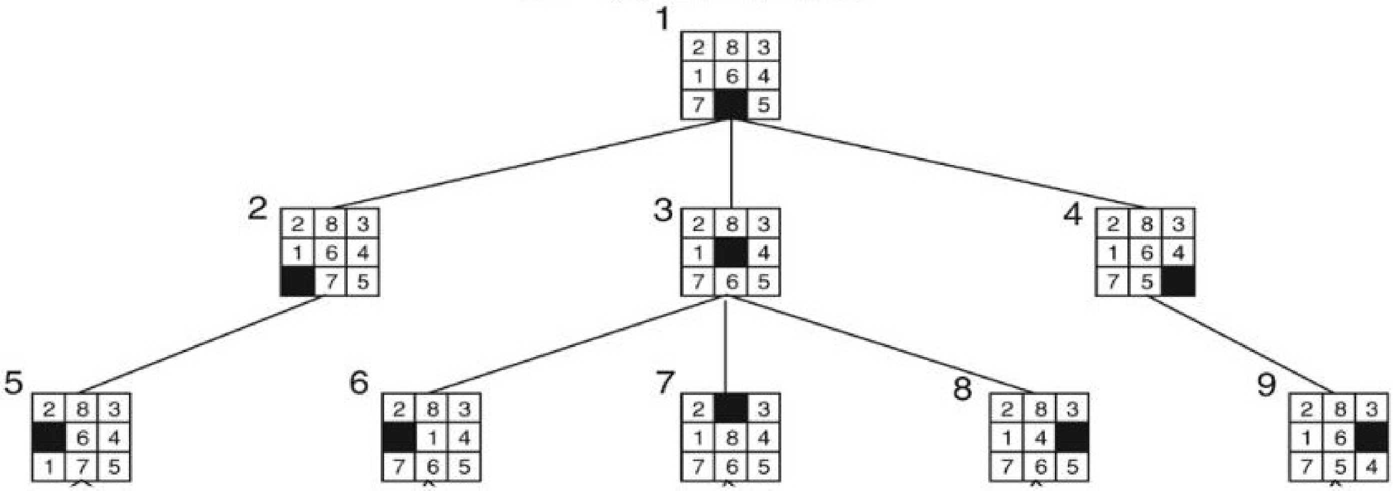
- Find all the possible states of the 15-Puzzle starting from the given initial state. Note that, the states should be unique (no repetitions).
- From the initial state of the puzzle, use different moves in all the directions to generate new states, check the validity of the newly generated node.
- Traverse the path as a tree and reach the final state
- Programming in python is preferred.
- Follow flow-chart for better intuition

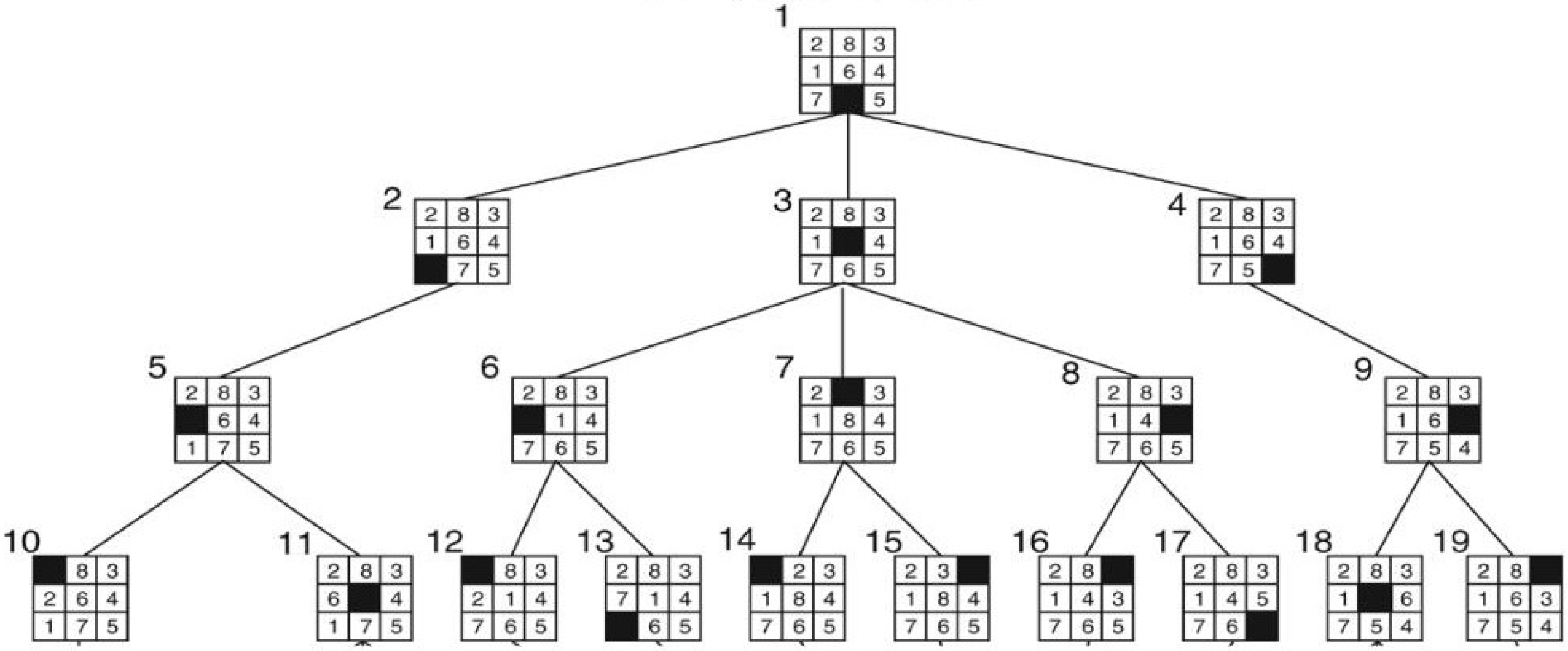
Example

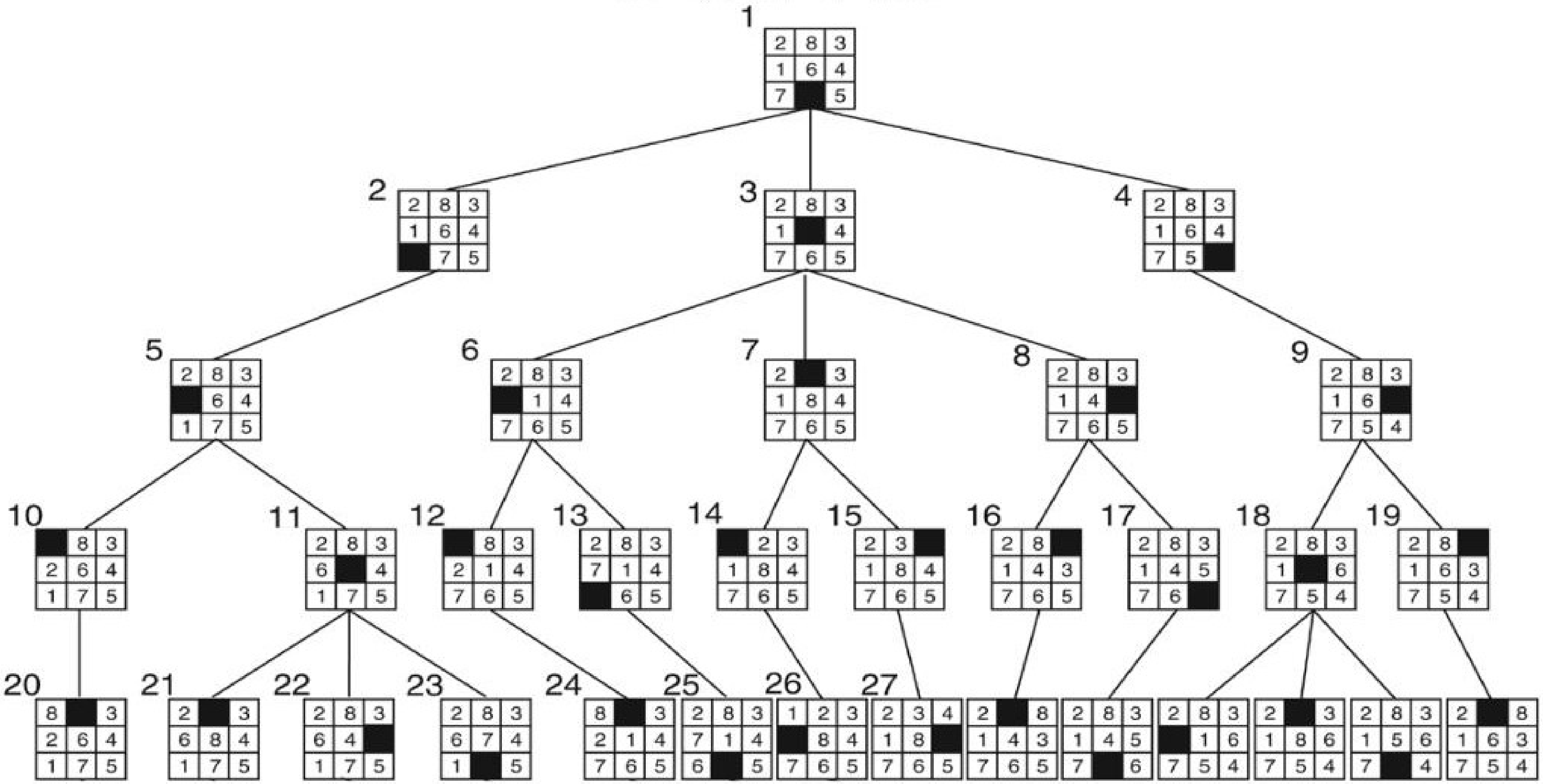


2	8	3
1	6	4
7	■	5

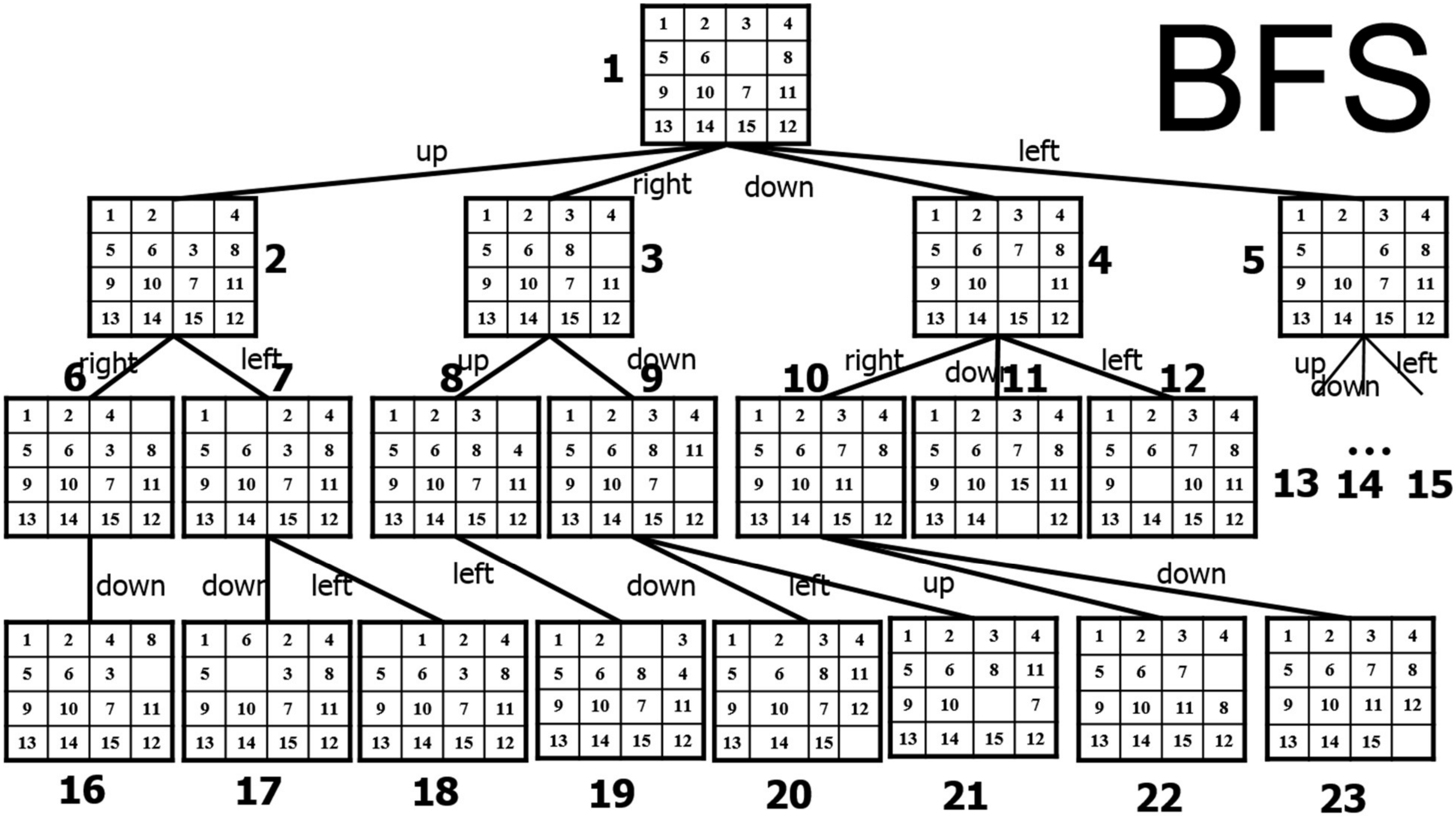

1
2 8 3
1 6 4
7 5
2
2 8 3
1 6 4
7 5
3
2 8 3
1 4
7 6 5
4
2 8 3
1 6 4
7 5


1
2 8 3
1 6 4
7 5
2
2 8 3
1 6 4
7 5
3
2 8 3
1 4
7 6 5
4
2 8 3
1 6 4
7 5
5
2 8 3
6 4
1 7 5
6
2 8 3
1 4
7 6 5
7
2 3
1 8 4
7 6 5
8
2 8 3
1 4
7 6 5
9
2 8 3
1 6
7 5 4


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19


1
2 8 3
1 6 4
7 5
2
2 8 3
1 6 4
7 5
3
2 8 3
1 4
7 6 5
4
2 8 3
1 6 4
7 5
5
2 8 3
6 4
1 7 5
6
2 8 3
1 4
7 6 5
7
2 3
1 8 4
7 6 5
8
2 8 3
1 4
7 6 5
9
2 8 3
1 6
7 5 4
10
8 3
2 6 4
1 7 5
11
2 8 3
6 4
1 7 5
12
8 3
2 1 4
7 6 5
13
2 8 3
7 1 4
6 5
14
2 3
1 8 4
7 6 5
15
2 3
1 8 4
7 6 5
16
2 8
1 4 3
7 6 5
17
2 8 3
1 4 5
7 6
18
2 8 3
1 6
7 5 4
19
2 8
1 6 3
7 5 4
20
8 3
2 6 4
1 7 5
21
2 3
6 8 4
1 7 5
22
2 8 3
6 4
1 7 5
23
2 8 3
6 7 4
1 5
24
8 3
2 1 4
7 6 5
25
2 8 3
7 1 4
6 5
26
1 2 3
8 4
7 6 5
27
2 3 4
1 8
7 6 5
2 8
1 4 3
7 6 5
2 8 3
1 4 5
7 6
2 8 3
1 6
7 5 4
2 3
1 8 6
7 5 4
2 8 3
1 5 6
7 4
2 8
1 6 3
7 5 4

BFS



1

1	2	3	4
5	6		8
9	10	7	11
13	14	15	12

2

1	2		4
5	6	3	8
9	10	7	11
13	14	15	12

3

1	2	3	4
5	6	8	
9	10	7	11
13	14	15	12

4

1	2	3	4
5	6	7	8
9	10		11
13	14	15	12

5

1	2	3	4
5		6	8
9	10	7	11
13	14	15	12

6

1	2	4	
5	6	3	8
9	10	7	11
13	14	15	12

7

1		2	4
5	6	3	8
9	10	7	11
13	14	15	12

8

1	2	3	
5	6	8	4
9	10	7	11
13	14	15	12

9

1	2	3	4
5	6	8	11
9	10	7	
13	14	15	12

10

1	2	3	4
5	6	7	8
9	10	11	
13	14	15	12

11

1	2	3	4
5	6	7	8
9	10	15	11
13	14		12

12

1	2	3	4
5	6	7	8
9		10	11
13	14	15	12

1	2	4	8
5	6	3	
9	10	7	11
13	14	15	12

16

1	6	2	4
5		3	8
9	10	7	11
13	14	15	12

17

	1	2	4
5	6	3	8
9	10	7	11
13	14	15	12

18

1	2		3
5	6	8	4
9	10	7	11
13	14	15	12

19

1	2	3	4
5	6	8	11
9	10	7	12
13	14	15	

20

1	2	3	4
5	6	8	11
9	10		7
13	14	15	12

21

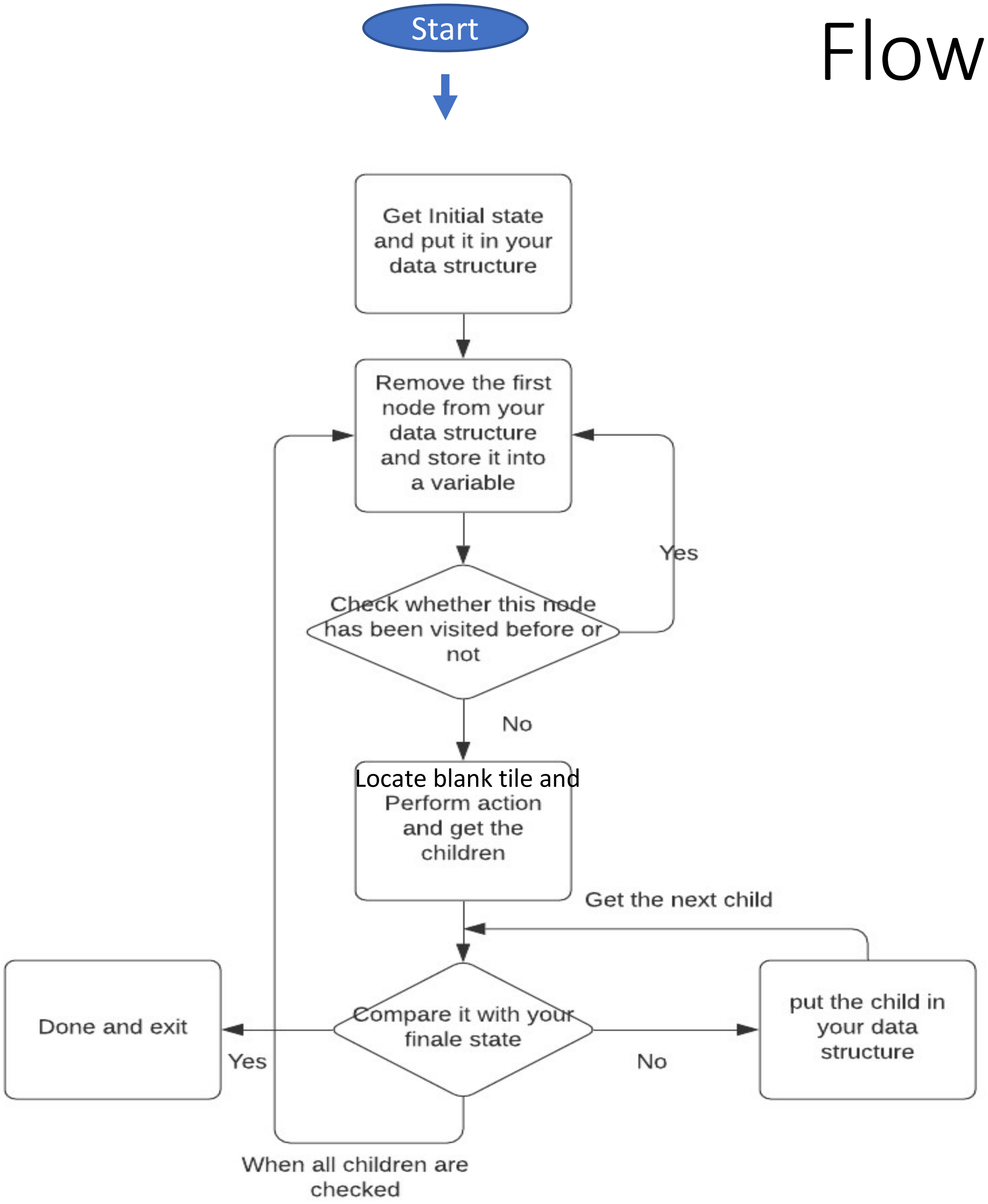
1	2	3	4
5	6	7	
9	10	11	8
13	14	15	12

22

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

23

Flow chart



Start

Get Initial state and put it in your data structure

Remove the first node from your data structure and store it into a variable

Check whether this node has been visited before or not

Yes

No

Locate blank tile and Perform action and get the children

Get the next child

Compare it with your finale state

Yes

Done and exit

No

put the child in your data structure

When all children are checked

Suggested Parameter Names

HINT:

Information to be stored in the data structure for each node:

- Node_State_i: The state of node i is represented by a 4 by 4 matrix, for example [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 0].

1-First Step

- You should first select a data structure type to save your generated nodes.
- There are different types of data structure that could be used. The type of data structure is optional.
- Hint: Read about stacks and queues

Data structure in PYTHON

Review the “list” and “dictionary” data structures for python in the following link

1) Learn Python - Full Course for Beginners [Tutorial] (Links to an external site.)

2) Learn Python - Full Course for Beginners [Tutorial] (Links to an external site.)
3) Learn Python - Full Course for Beginners [Tutorial]

4) https://www.linkedin.com/learning/learning-python-2 (Links to an external site.)

5) https://www.linkedin.com/learning/programming-foundations-data-structures-2 (Links to an external site.)

6) https://www.linkedin.com/learning/python-data-structures-stacks-queues-and-deques

2. Step two

- Write a function that given the state of the current node in the 15 puzzle problem, calculates the location of the blank tile in the 4 by 4 matrix and returns the output as a pair, (i,j). Where 0<i<4 and 0<j<4.
- Blanktile can be located using either two for loops or by using inbuilt functions of numpy.

% Find the location of the blank tile to take further actions

3. Step three

- Write 4 subfunctions to move the blank tile in 4 different directions and store them in a list

```
[NewNode] = ActionMoveLeft(CurrentNode);
% Moves blank tile left, if possible

[NewNode] = ActionMoveRight(CurrentNode);
% Moves blank tile right, if possible

[NewNode] = ActionMoveUp(CurrentNode);
% Moves blank tile up, if possible

[NewNode] = ActionMoveDown(CurrentNode);
% Moves blank tile down, if possible
```

4. Step four

- Append All the possible new nodes in a list
- Follow the flowchart (optional) once done

- Additional: Focus on checking the node to be explored with the visited list you maintain and if it is present do not repeat the actionset for that node.

Additional Notes

- Parent child information is optional and not mandatory. No extra marks will be cut if the parent information is not stored with the current node.

- Each test case will be allowed to run for a maximum of two hours to be considered for evaluation, i.e. given an initial node, the fifteen puzzle should be solved within this time.
- Randomly 10 students will be selected who need to explain their code during TA office hours.

Due Date and Deliverables

- Due date: 22nd Feb 11:59 pm .
- Submit deliverables on Canvas & GitHub.
- Deliverables:
  - Source code (Should be a python file)
  - Output textfile to be generated from the code: Nodes
  - A text file that explains how to run the program.
- Folder Name (zip) : proj1_firstname_lastname

Additional Hint

- Since it’s 15 puzzle for some test cases it might take a long time to reach the goal state thus it’s important to consider the code efficiency.
- Increasing Efficiency:
  - Use a Good data structure (For Example: List, dictionary, Queue (Refer to the links)
  - Store Matrix as an integer or string which will help while comparing to different nodes.
  - While performing action read the integer/string and store it as a matrix for ease. (optional)

GitHub and README

- GitHub
  - Well commented code
  - Frequent detailed commits, i.e. commits should have messages. For example - “ Added function to find the blank tile location”
  - The link (made public) should be uploaded as comments with your submission.
- ReadMe
  - Instructions to run the code, step by step.
  - Explicitly mention the libraries used in your code

Output textfile to be generated from the code

Textfile 1:

Name: nodePath.txt

The elements are being stored column-wise, i.e. for this state 1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 0 , the fifteen-puzzle state is

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

The order of the states should be from start node to the goal node.