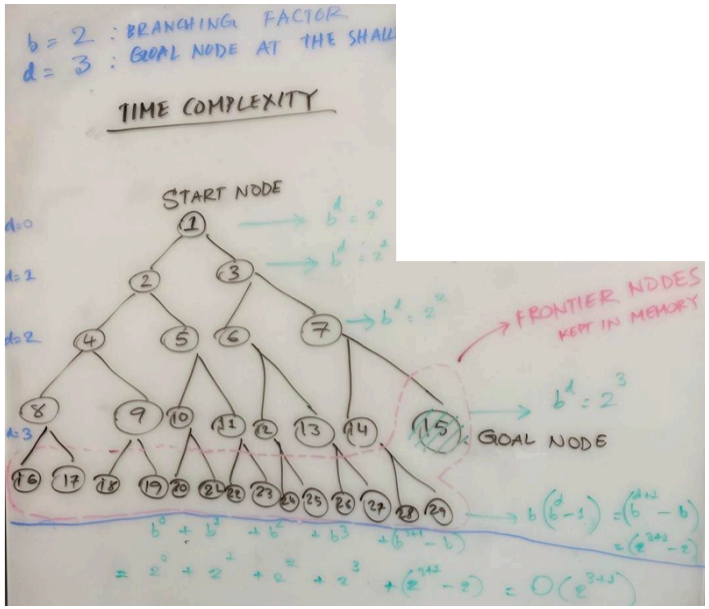
CS143: Artificial Intelligence

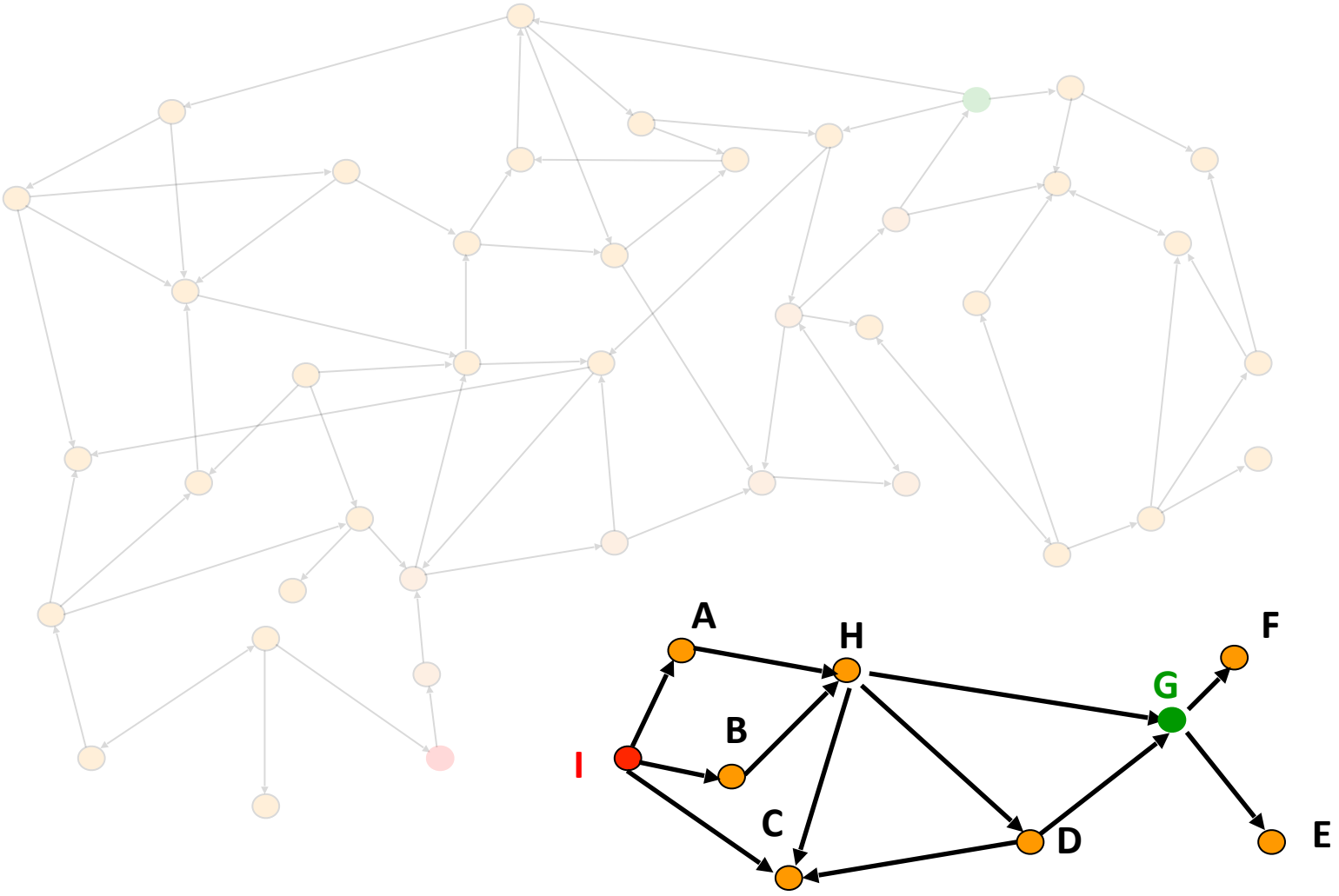


BFS and DFS implementation for **mapbot**

Analysis of Search Algorithms (BFS, DFS, IDS)

Drake UNIVERSITY

Recap: Exercise - BFS on a Graph with loop



Recap: Iterative Deepening Search (IDS)

- Uses a limited version of DFS called depth-limited search
  - nodes at depth l has no successors

- do a depth-limited search with l = 0, if it fails,
- do a depth-limited search with l = 1, if it fails,
- do a depth-limited search with l = 2, if it fails,

  …

  …

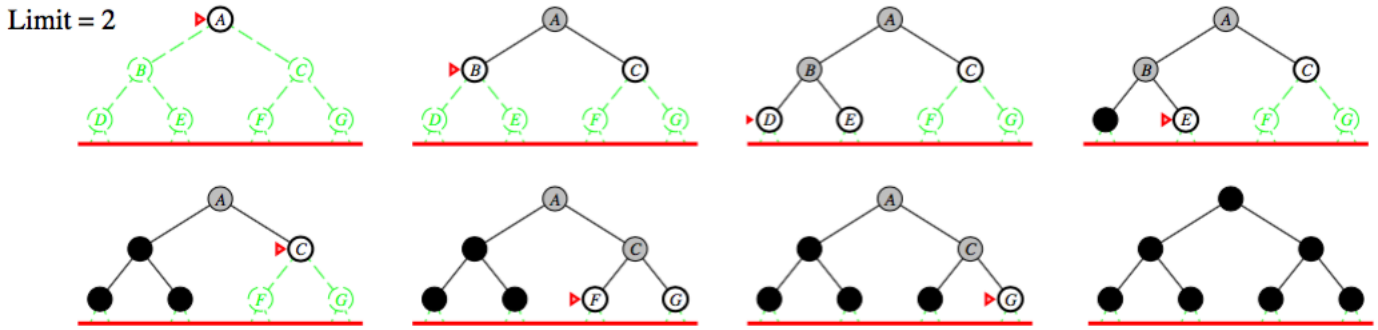
  …
- until you find a goal

Recap: Iterative Deepening Search (IDS)

Limit = 0

Recap: Iterative Deepening Search (IDS)

Recap: Iterative Deepening Search (IDS)



Today: Practical Implementation

- Practical implementation of AI problem solving as search
  - Introduced the python code for search last week, only spent last few minutes on that
  - Explore the code in-class and try to solve the activity together

Analysis of BFS

Properties of Breadth-First Search (BFS)

- **Completeness:**
  - a search algorithm is **complete** if it finds a solution whenever one exists
- **Optimality:**
  - a search algorithm is **optimal** if it returns a minimum-cost path if a solution exists
- **Time complexity:**
  - time required by the algorithm
- **Space complexity:**
  - memory required by the algorithm

Properties of Breadth-First Search (BFS)

- Is BFS Complete?
- Is BFS Optimal?
- What is the running time complexity of BFS?
- What are the memory requirement of BFS?

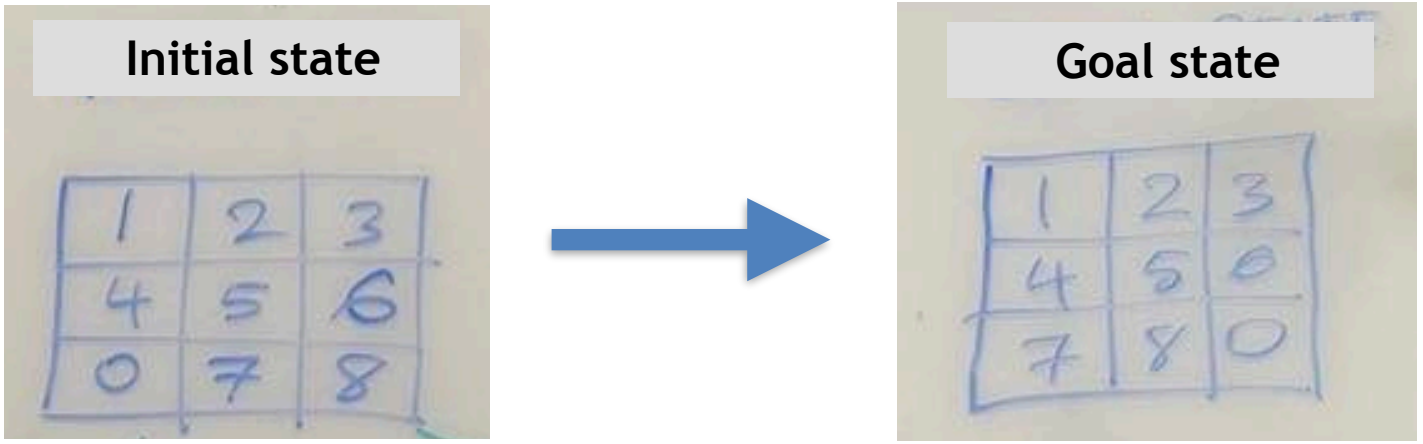
Properties of Breadth-First Search (BFS)

b: branching factor
d: depth of shallowest goal node

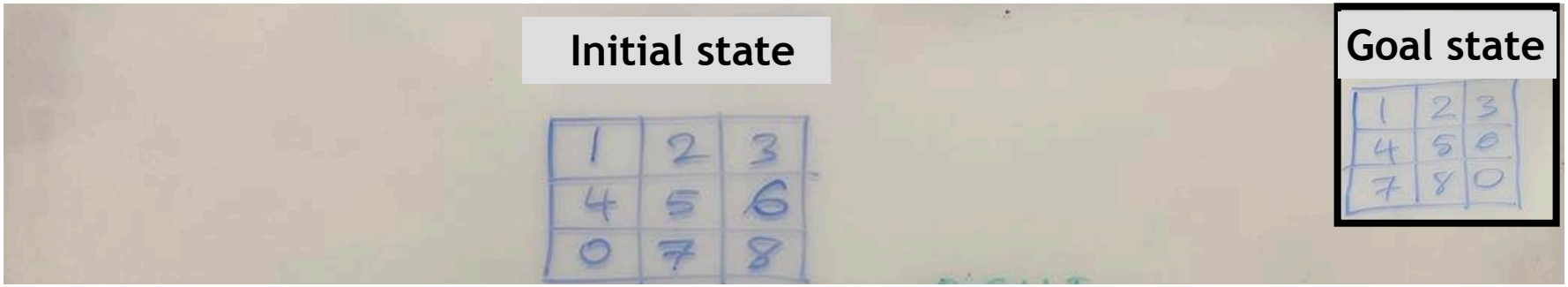
- Is BFS complete?
  - BFS is complete for a finite branching factor b

- Is BFS optimal?
  - BFS is optimal if step cost is 1
  - BFS will find the goal node at the shallowest depth before reaching goal nodes at a deeper level

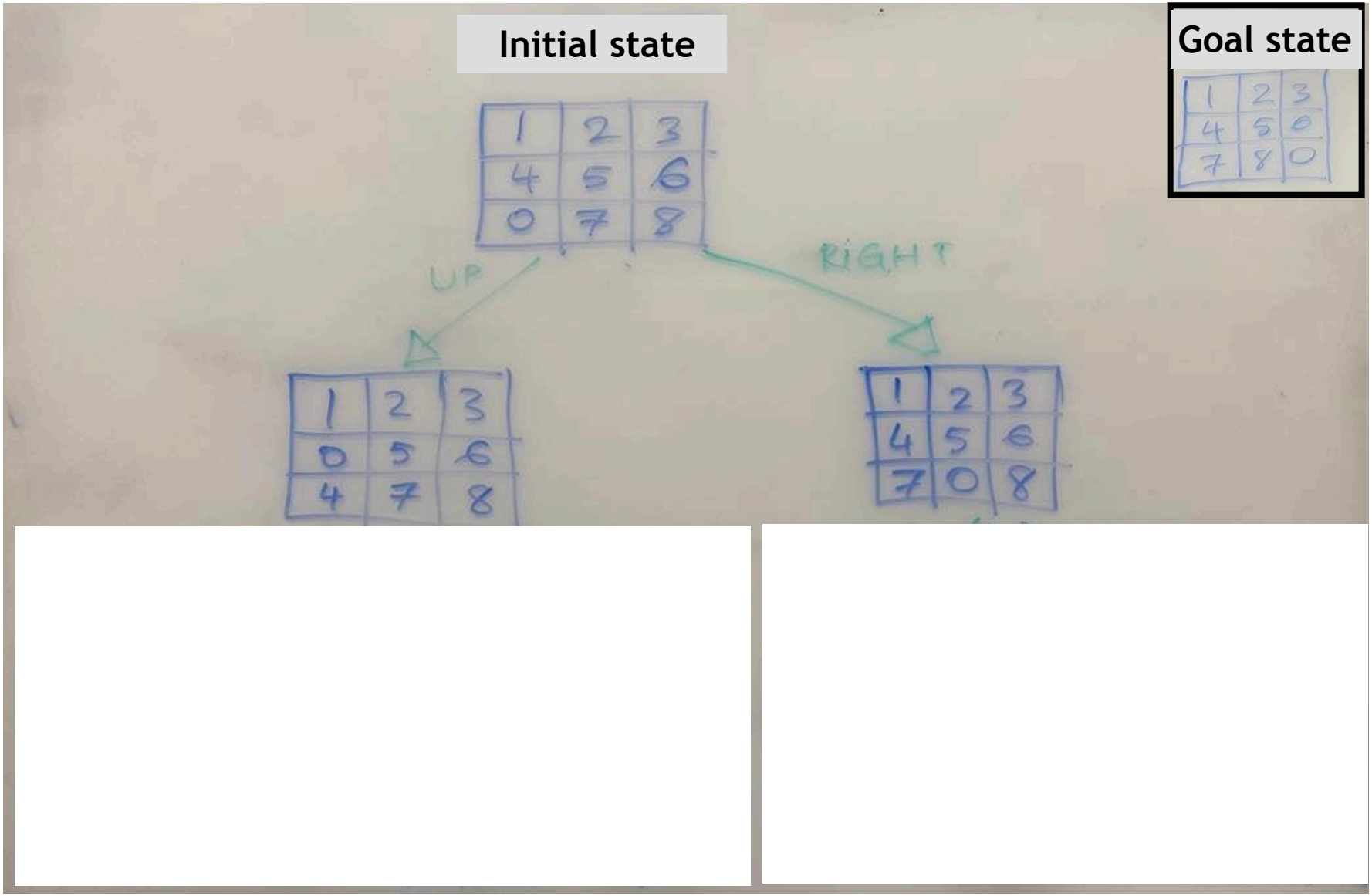
Let's Analyze 8-puzzle with BFS



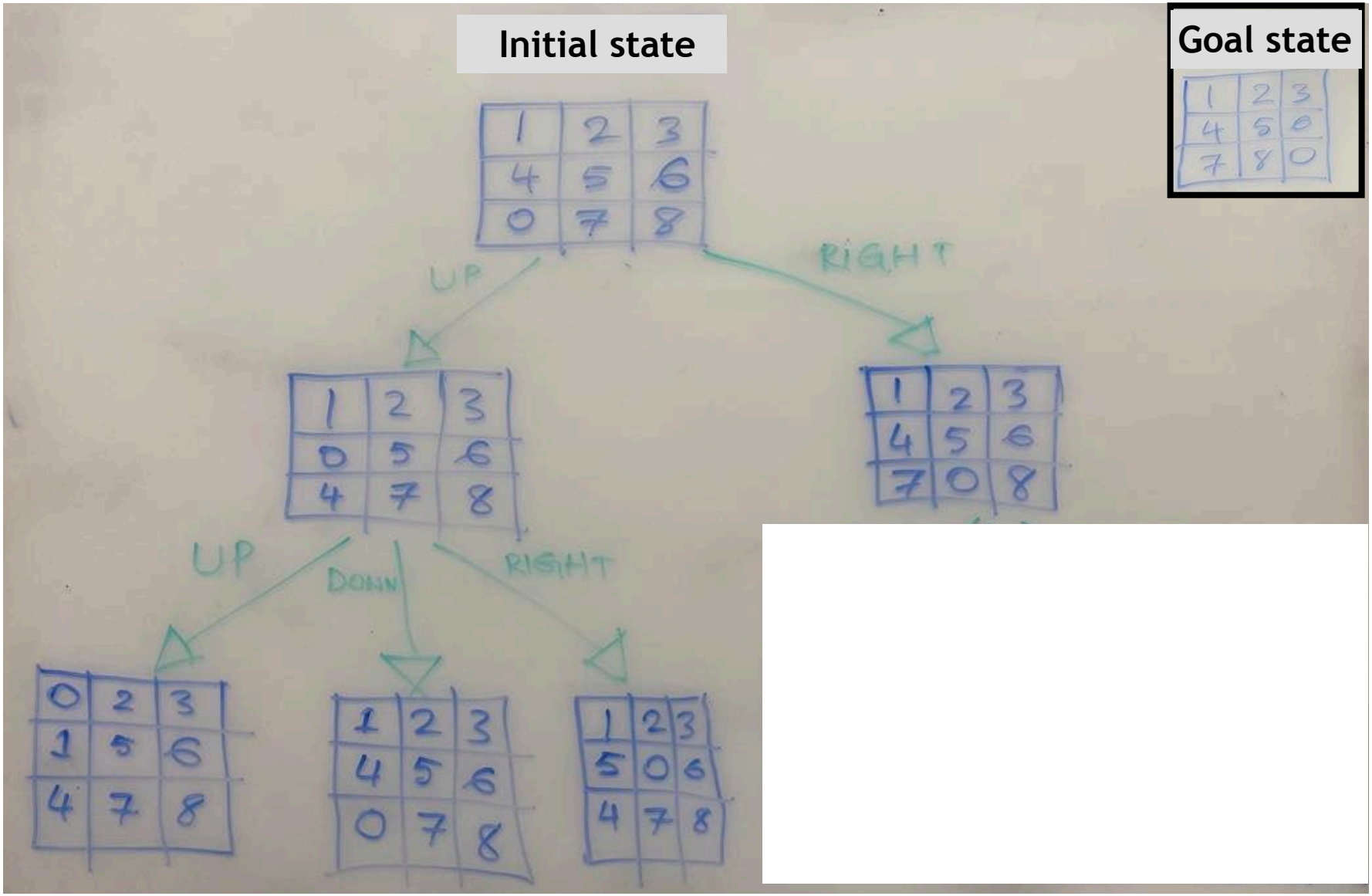
Let's Analyze 8-puzzle with BFS



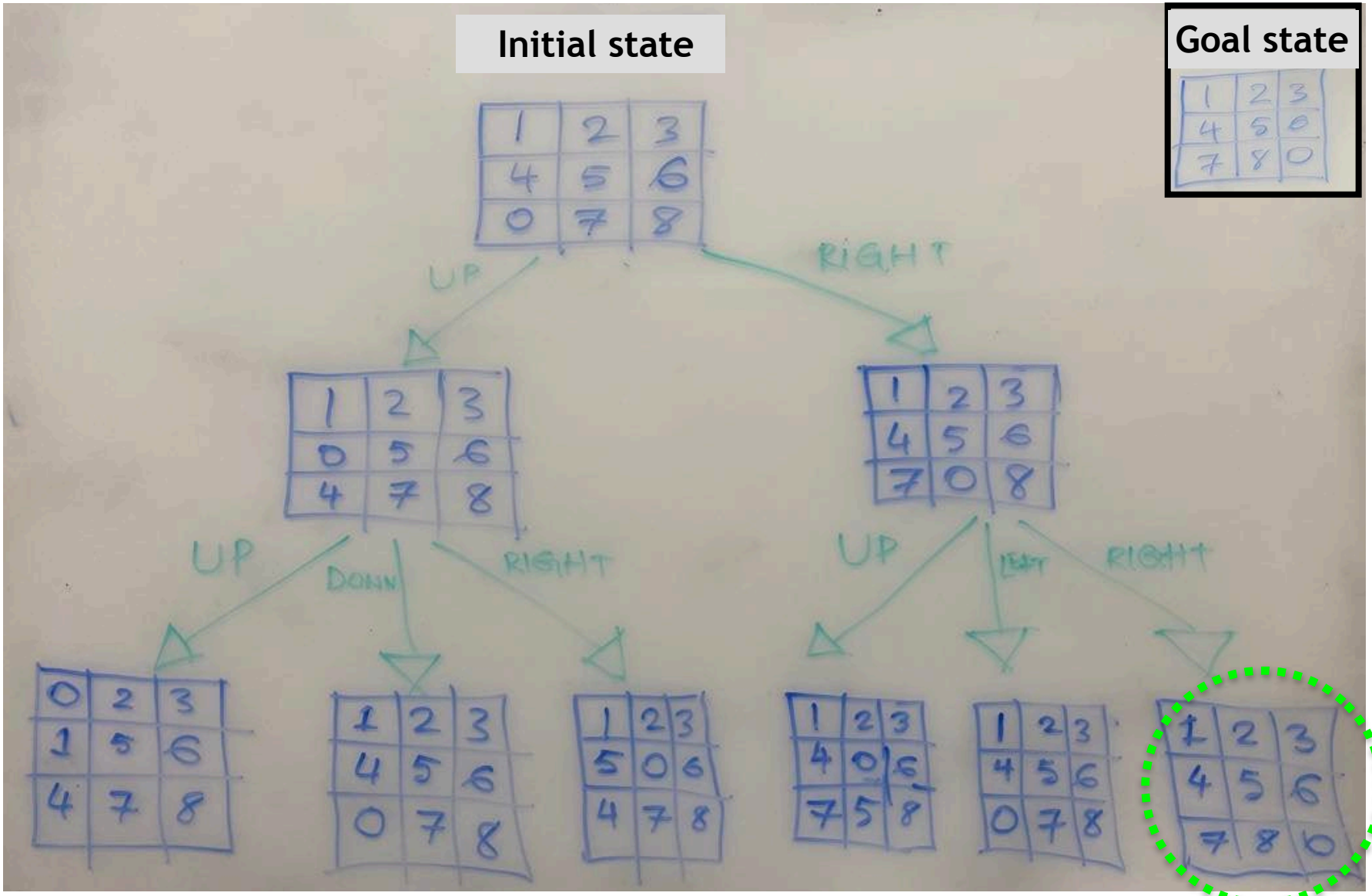
Let's Analyze 8-puzzle with BFS



Let's Analyze 8-puzzle with BFS



Let's Analyze 8-puzzle with BFS



Properties of Breadth-First Search (BFS)

b: branching factor
d: depth of shallowest goal node

- Time complexity of BFS?

  Number of nodes generated (in the worst-case):

  $1 + b + b^2 + \ldots + b^d + b*(b^d - 1) = O(b^{d+1})$

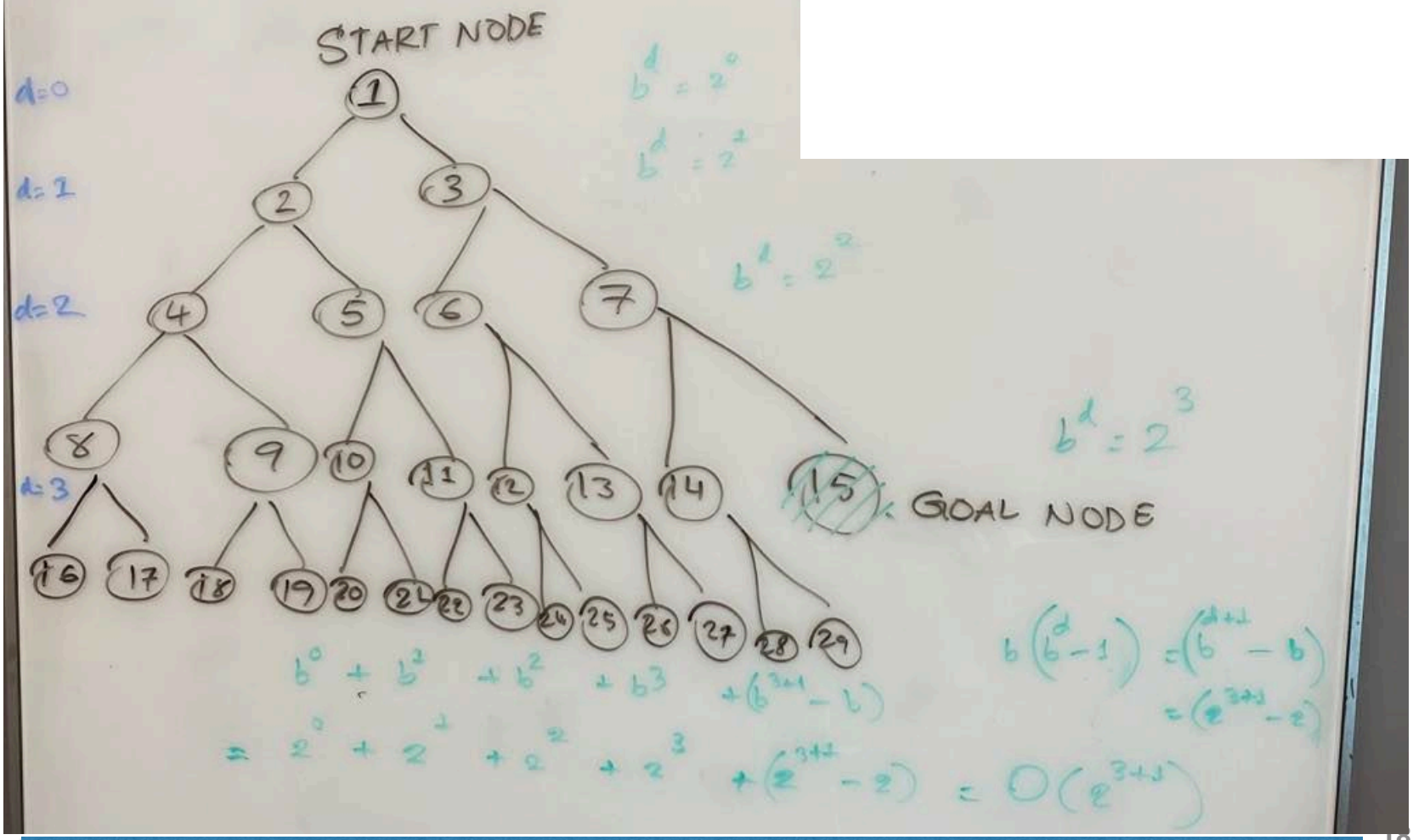
  Time complexity is $O(b^{d+1})$

- Let's check — how did we arrive at above formula.

$b = 2$: BRANCHING FACTOR

$d = 3$: GOAL NODE AT THE SHALLOWEST DEPTH

TIME COMPLEXITY



Properties of Breadth-First Search (BFS)

- Time is a big concern for BFS
  - If the problem has a solution at depth 16 with branching factor 10, it may take about 350 years for BFS to find it

d	# Nodes	Time
2	111	0.11 msec
4	11,111	11.11 msec
6	~10^6	1 sec
8	~10^8	100 sec
10	~10^{10}	2.8 hours
12	~10^{12}	11.6 days
14	~10^{14}	3.2 years

Assumptions:
- b = 10
- A computer can process 1,000,000 nodes/sec

Properties of Breadth-First Search (BFS)

b: branching factor
d: depth of shallowest goal node

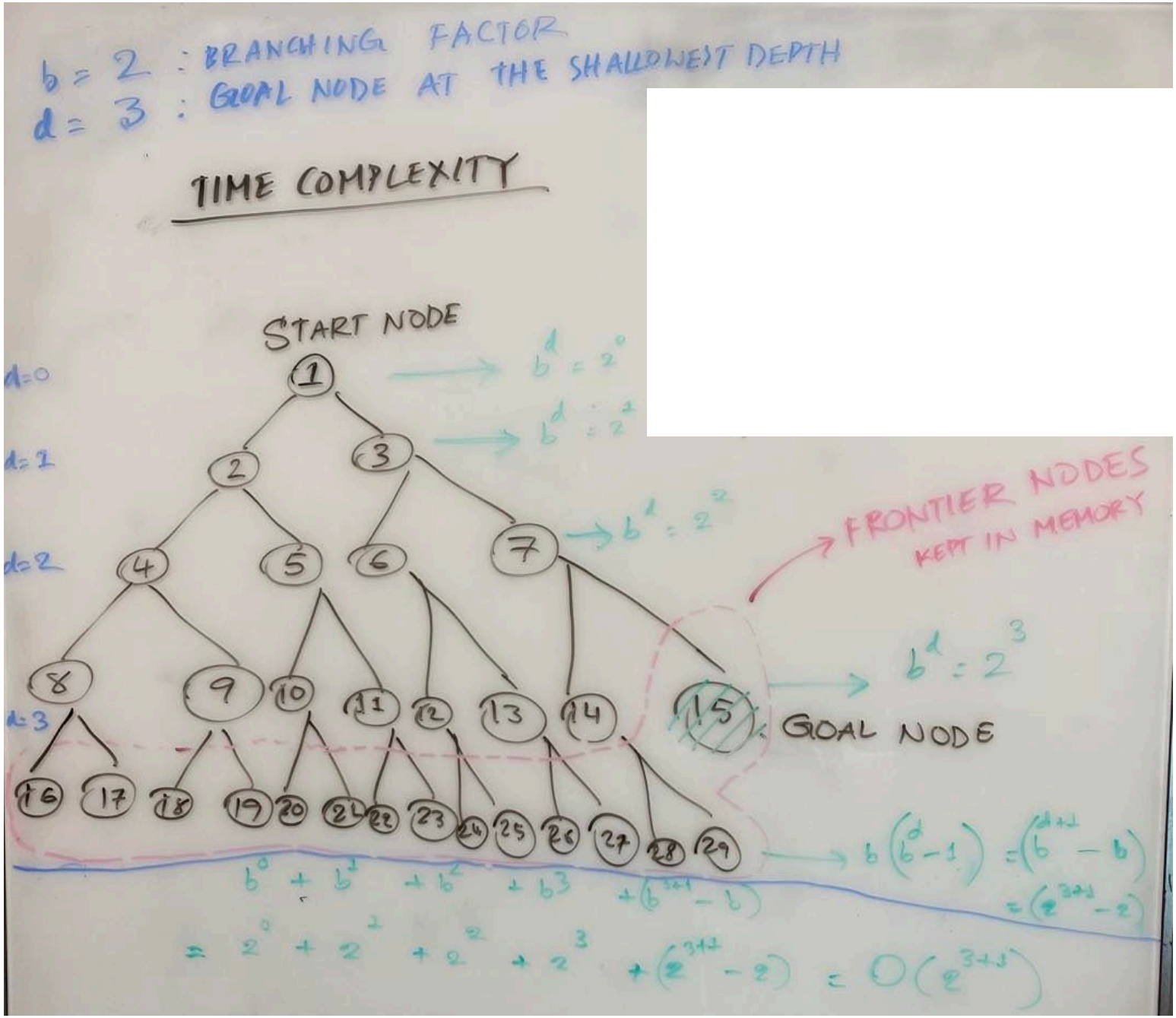
- Memory complexity of BFS?

  Number of nodes kept in the frontier

  $b*(b^d - 1) = O(b^{d+1})$

  Time and space complexity is $O(b^{d+1})$

- Memory is a big concern for BFS.
  - can easily generate nodes at 100MB/second so gigabytes of space required if BFS runs for hours


b = 2 : BRANCHING FACTOR
d = 3 : GOAL NODE AT THE SHALLOWEST DEPTH
TIME COMPLEXITY
START NODE
d=0
d=1
d=2
d=3
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
FRONTIER NODES KEPT IN MEMORY
GOAL NODE
$b^d = 2^0$
$b^d = 2^1$
$b^d = 2^2$
$b^d = 2^3$
$b^0 + b^1 + b^2 + b^3 + (b^{d+1} - b)$
$= 2^0 + 2^1 + 2^2 + 2^3 + (2^{d+1} - 2) = O(2^{d+1})$
$b\left(\frac{b^d - 1}{b-1}\right) = \left(\frac{b^{d+1} - b}{b-1}\right)$
$= (2^{d+1} - 2)$

Properties of Breadth-First Search (BFS)

- Time is a big concern for BFS
  - If the problem has a solution at depth 16 with branching factor 10, it may take about 350 years for BFS to find it

d	# Nodes	Time	Memory
2	111	0.11 msec	11 Kbytes
4	11,111	11.11 msec	1 Mbyte
6	~10^6	1 sec	100 Mb
8	~10^8	100 sec	10 Gbytes
10	~10^{10}	2.8 hours	1 Tbyte
12	~10^{12}	11.6 days	100 Tbytes
14	~10^{14}	3.2 years	10,000 Tbytes

Assumptions:
- b = 10
- A computer can process 1,000,000 nodes/sec

Analysis of DFS

Properties of Depth-First Search

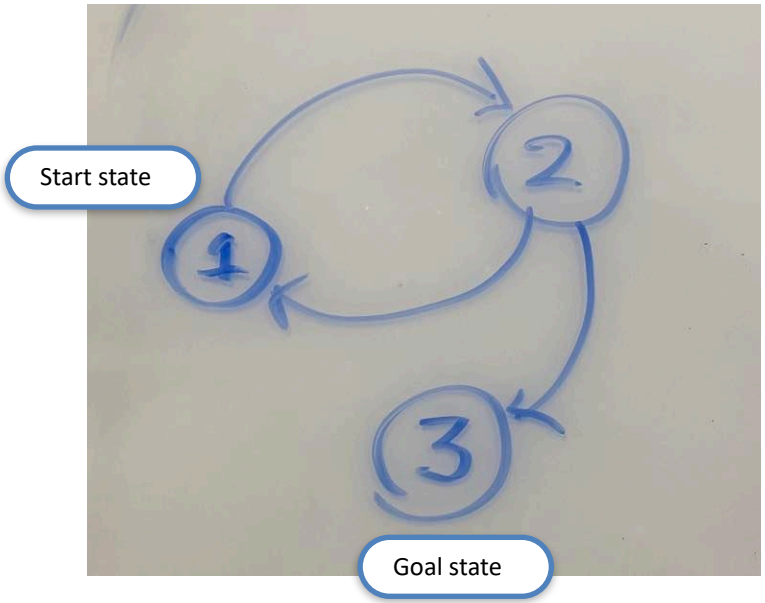
- **Completeness:**
  - a search algorithm is **complete** if it finds a solution whenever one exists
- **Optimality:**
  - a search algorithm is **optimal** if it returns a minimum-cost path if a solution exists
- **Time complexity:**
  - time required by the algorithm
- **Space complexity:**
  - memory required by the algorithm

Properties of Depth-First Search

- Is DFS Complete?
- Is DFS Optimal?
- What is the running time complexity of DFS?
- What are the memory requirement of DFS?

Properties of Depth-First Search

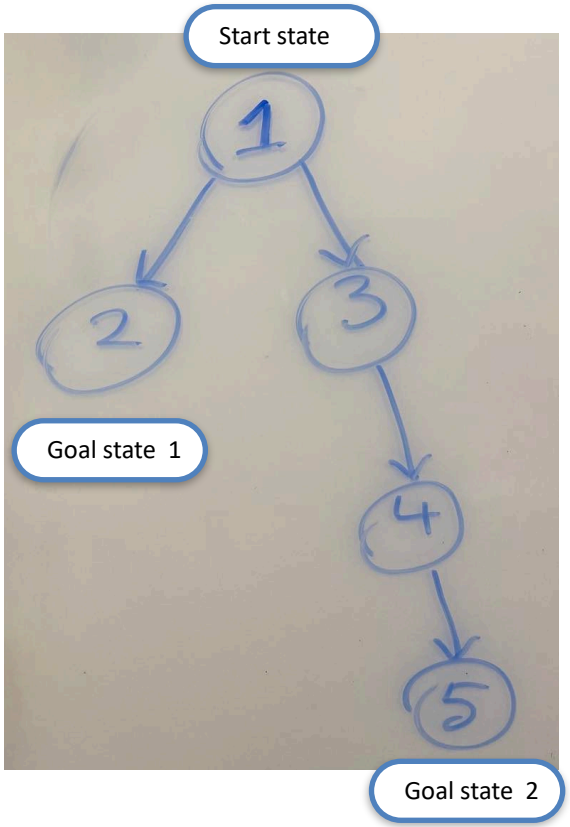
- Is DFS Complete?



- It is not complete even if the state graph is finite
  - The search tree could be infinite
  - Eg, [1]->[2]->[3, 1]->[3, 2]->[3, 3, 1]->[3, 3, 2]-> …

Properties of Depth-First Search

- Is DFS Optimal?



- It is not optimal
  - The search tree could not reach goal state at a lower depth
  - Eg, [1]->[2,3]->[2, 4]->[2, 5]

Properties of Depth-First Search

b: branching factor

l: depth of deepest leaf node

- Depth-first search
  - Not complete
  - Not optimal

- Number of nodes generated:

  $1 + b + b^2 + \ldots + b^l = O(b^l)$

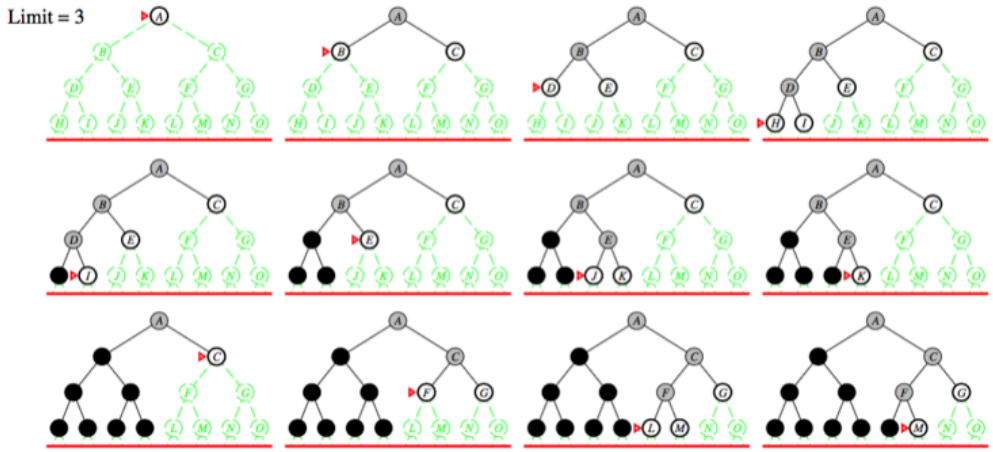
- Time and space complexity is $O(b*l)$

Analysis of IDS

Properties of Iterative Deepening Search (IDS)

- Is IDS Complete?
- What is the running time complexity of IDS?
- What are the memory requirement of IDS?
- Is IDS Optimal?

Properties of Iterative Deepening Search (IDS)



- IDS is complete
- IDS is optimal
- Number of nodes generated:

$$(d+1)b^0+db^1+(d-1)b^2+\ldots+b^d = O(b^d)$$

- Space complexity is $O(b*d)$

Number of Generated Nodes: BFS vs IDS

b: branching factor

d: depth of shallowest goal node

- b = 2 and d = 5

	BFS	IDS
Nodes at depth 1	1	1 x 6 = 6
Nodes at depth 2	2	2 x 5 = 10
Nodes at depth 3	4	4 x 4 = 16
Nodes at depth 4	8	8 x 3 = 24
Nodes at depth 5	16	16 x 2 = 32
Nodes at depth 5	32	32 x 1 = 32
	63	120

Number of Generated Nodes: BFS vs IDS

b: branching factor

d: depth of shallowest goal node

- b = 10 and d = 5

BFS	IDS
1	6
10	50
100	400
1,000	3,000
10,000	20,000
100,000	100,000
111,111	123,456

Project Assignment#2

- Implement BFS, DFS, and IDS on Mapbot
  - base code will be given; modify and append additional python snippets
  - answer questions

- Will be released later tonight!