8 Puzzle Problem

using

Greedy Best First Search

Course Instructor:- BIPLAB PALAYE

Group Members

ASHOK KUMAR
DIVYESH ANSHU

What is 8 Puzzle Problem?

The 8 puzzle consists of an area divided into a grid, 3 by 3 for the 8-puzzle . Thus, there are eight tiles in the 8-puzzle. A tile that is next to the empty grid square can be moved into the empty space, leaving its previous position empty in turn. Tiles are numbered, 1 to 8 for the 8-puzzle, so that each tile can be uniquely identified.

We have to solve it using **Greedy Best First Search**

Greedy Best First Search

Best-first search is a search algorithm which explores a graph by expanding the most promising node chosen according to a specified rule.

Greedy best-first search expands the node that appears to be closest to goal . Here **f(n) = h(n)** *{ where **h(n)** is the estimate of cost from node n to goal }*

Properties of Greedy best-first search

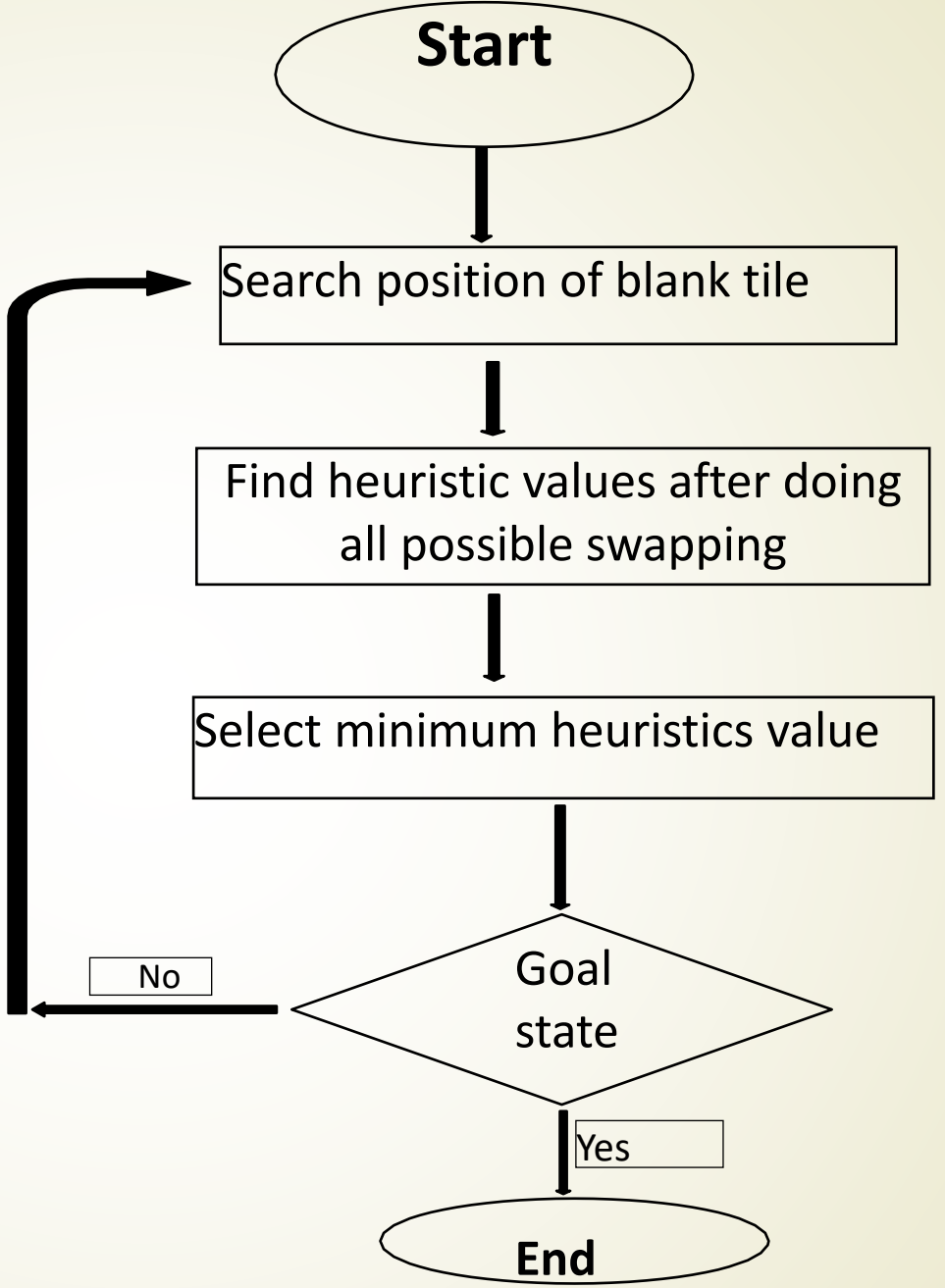
Complete ? No – It can get stuck in loops.

Time ? O(bm), but a good heuristic can give dramatic improvement

Space ? O(bm) - keeps all nodes in memory

Optimal ? No

FLOW CHART:


Start
Search position of blank tile
Find heuristic values after doing all possible swapping
Select minimum heuristics value
Goal state
No
Yes
End

Pseudo Code of Greedy BFS

```
Procedure: GreedyBFS
insert (state=initial_state, h=initial_heuristic, counter=0) into search_queue;

while search_queue not empty do
   current_queue_entry = pop item from front of search_queue;
   current_state = state from current_queue_entry;
   current_heuristic = heuristic from current_queue_entry;
   starting_counter = counter from current_queue_entry;
   applicable_actions = array of actions applicable in current_state;

   for all index ?i in applicable_actions ≥ starting_counter do
      current_action = applicable_actions[?i];
      successor_state = current_state.apply(current_action);

      if successor_state is goal then
         return plan and exit;
      end if
      successor_heuristic = heuristic value of successor_state;

      if successor_heuristic < current_heuristic then
         insert (current_state, current_heuristic, ?i + 1) at front of search_queue;
         insert (successor_state, successor_heuristic, 0) at front of search_queue;
         break for;

      else
         insert (successor_state, successor_heuristic, 0) into search_queue;
      end if
   end for
end while
exit - no plan found;
```

Figure Shows the initial and goal state of the **8 Puzzle** Problem. Solve this Puzzle using **Greedy BFS**

Initial State

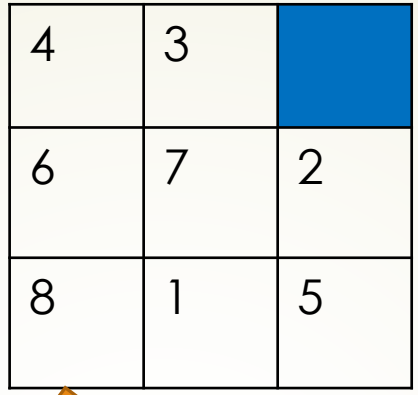
4	3	
6	7	2
8	1	5

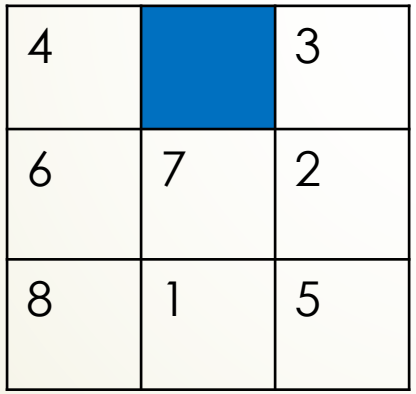
Goal State

	1	2
3	4	5
6	7	8

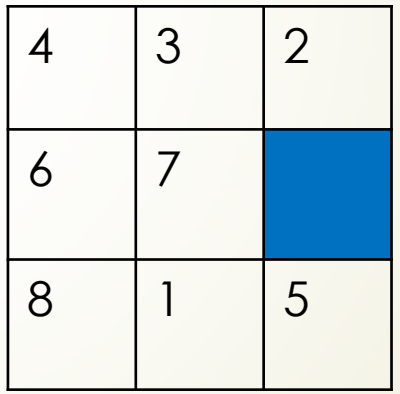
Here the heuristic function **h(n)** is the no. of misplaced tiles (*excluding the blank tile*) with respect to the goal position. We need to solve the above problem using **Greedy Best First Search** algorithm

Initially h(n)=8





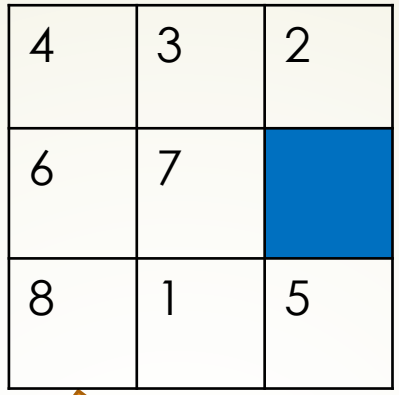
Move left h(n)=8

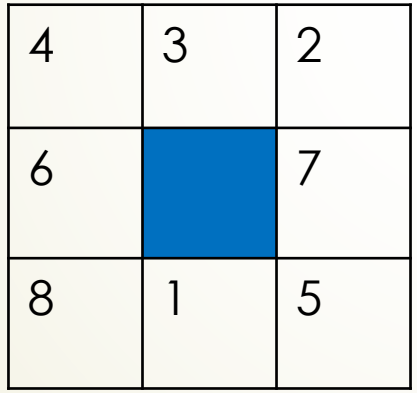


Move down h(n)=7

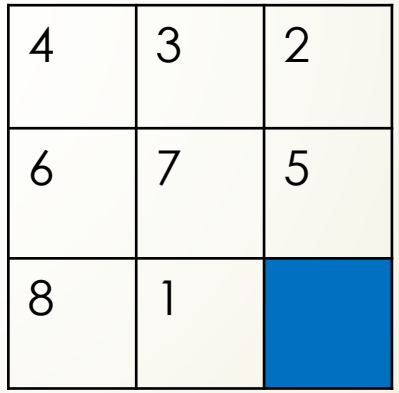
So here the block will move down

Now h(n)=7





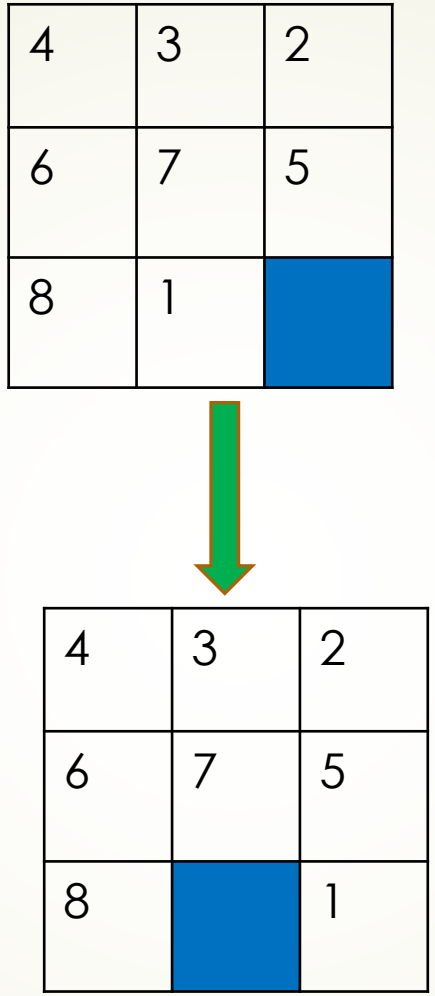
Move left ; h(n)=7



Move down ; h(n)=6

So here the block will move down

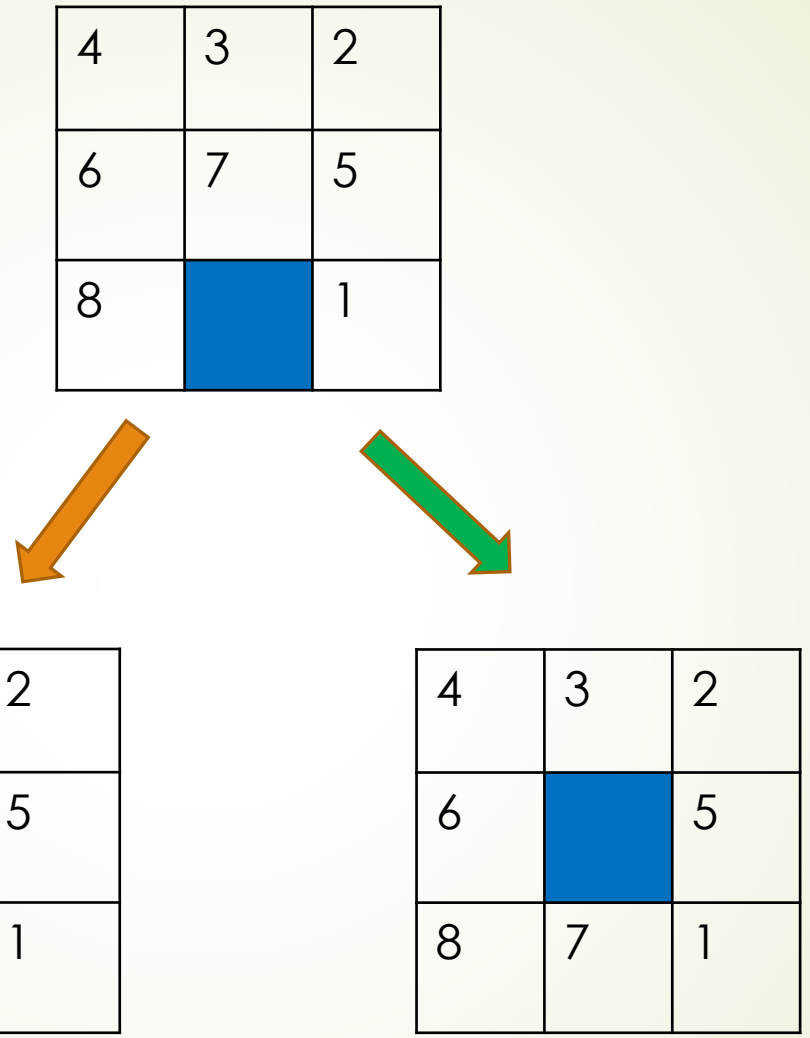
Now h(n)=6



Move left ; h(n)=6

So here the block will move left

Now h(n)=6

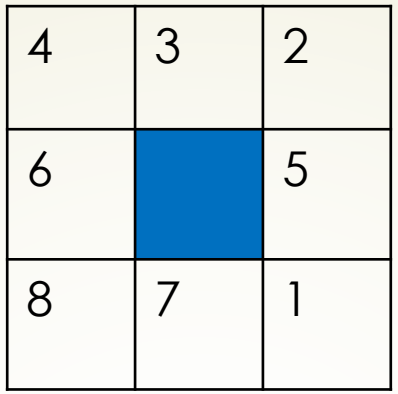


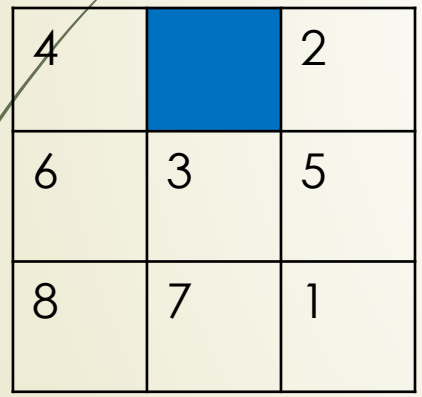
Move left ; h(n)=6

Move up ; h(n)=5

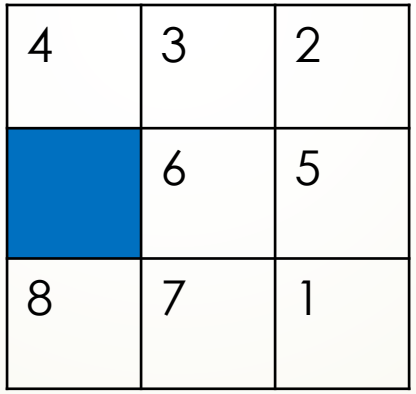
Here we will move the block up

Initially h(n)=6

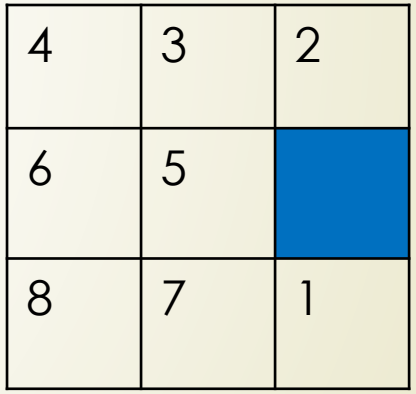




Move up ; h(n)=5



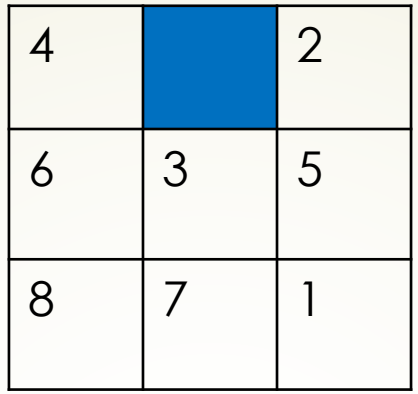
Move left ; h(n)=5

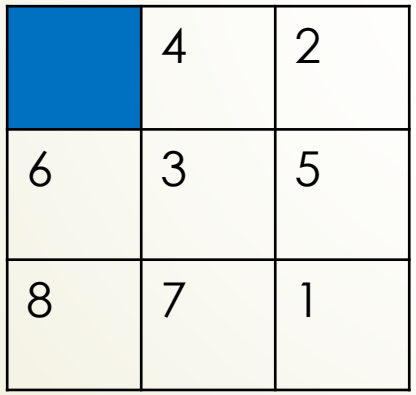


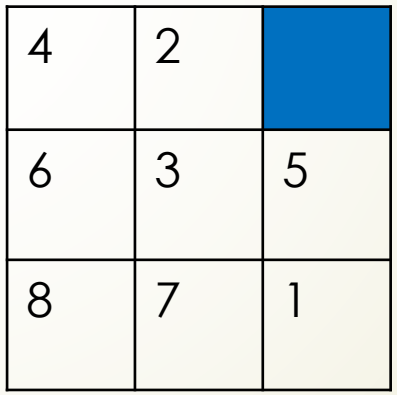
Move right ; h(n)=6

Here we are getting equal value of heuristic function for two moves , so we will randomly choose any one of the move , lets say UP .

Now h(n)=5







Move left ; h(n)=5

Move right ; h(n)=6

Here the block will move left

Now h(n)=5

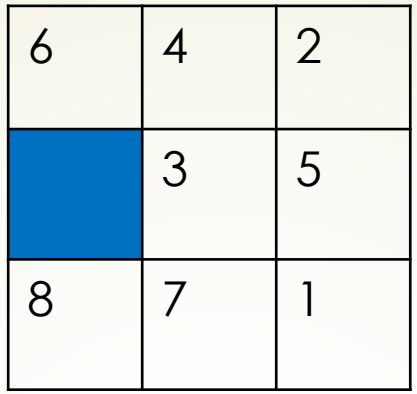
	4	2
6	3	5
8	7	1

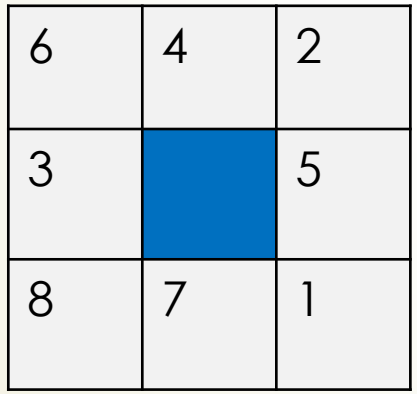
6	4	2
	3	5
8	7	1

Move down ; h(n)=5

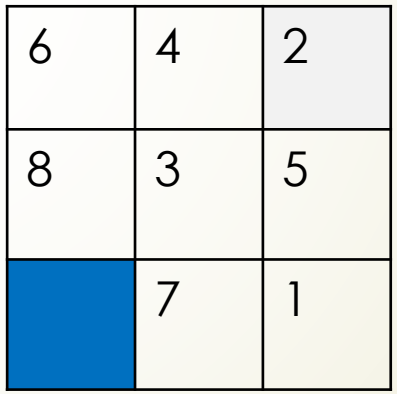
Here the block will move down

Now h(n)=5





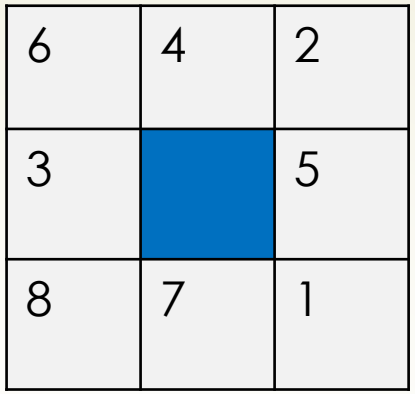
Move right; h(n)=4

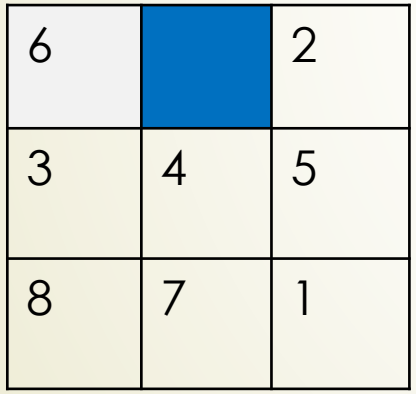


Move down ; h(n)=5

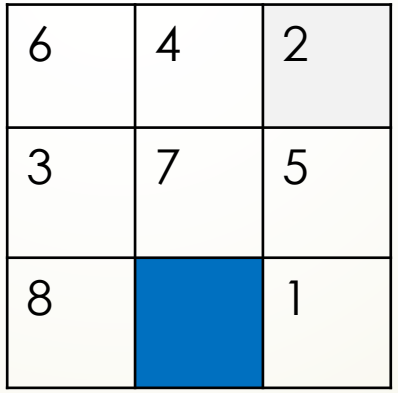
Here the block will move right

Now h(n)=4

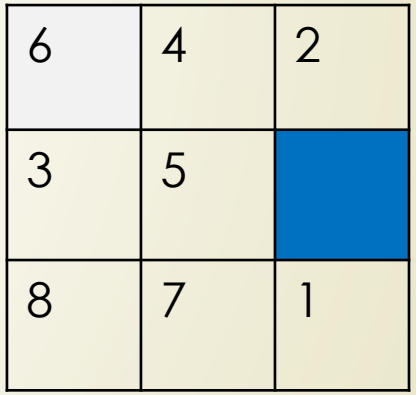




Move up h(n)=4



Move down ; h(n)=5



Move right ; h(n)=5

Here the block will move up

Now h(n)=4

6		2
3	4	5
8	7	1

	6	2
3	4	5
8	7	1

Move left h(n)=3

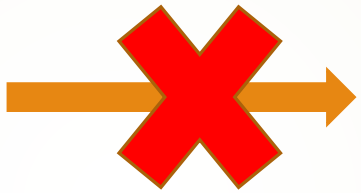
6	2	
3	4	5
8	7	1

Move right ; h(n)=5

Here the block will move left

Now h(n)=3

	6	2
3	4	5
8	7	1



3	6	2
	4	5
8	7	1

Move down h(n)=4

Now from here if we try to move the block down, the value of heuristic function becomes 4 , and so we have no option to move anywhere as Greedy BFS Does not allows us to do so. So this problem cannot be solved completely.

Let's take a look at another example

Example:-

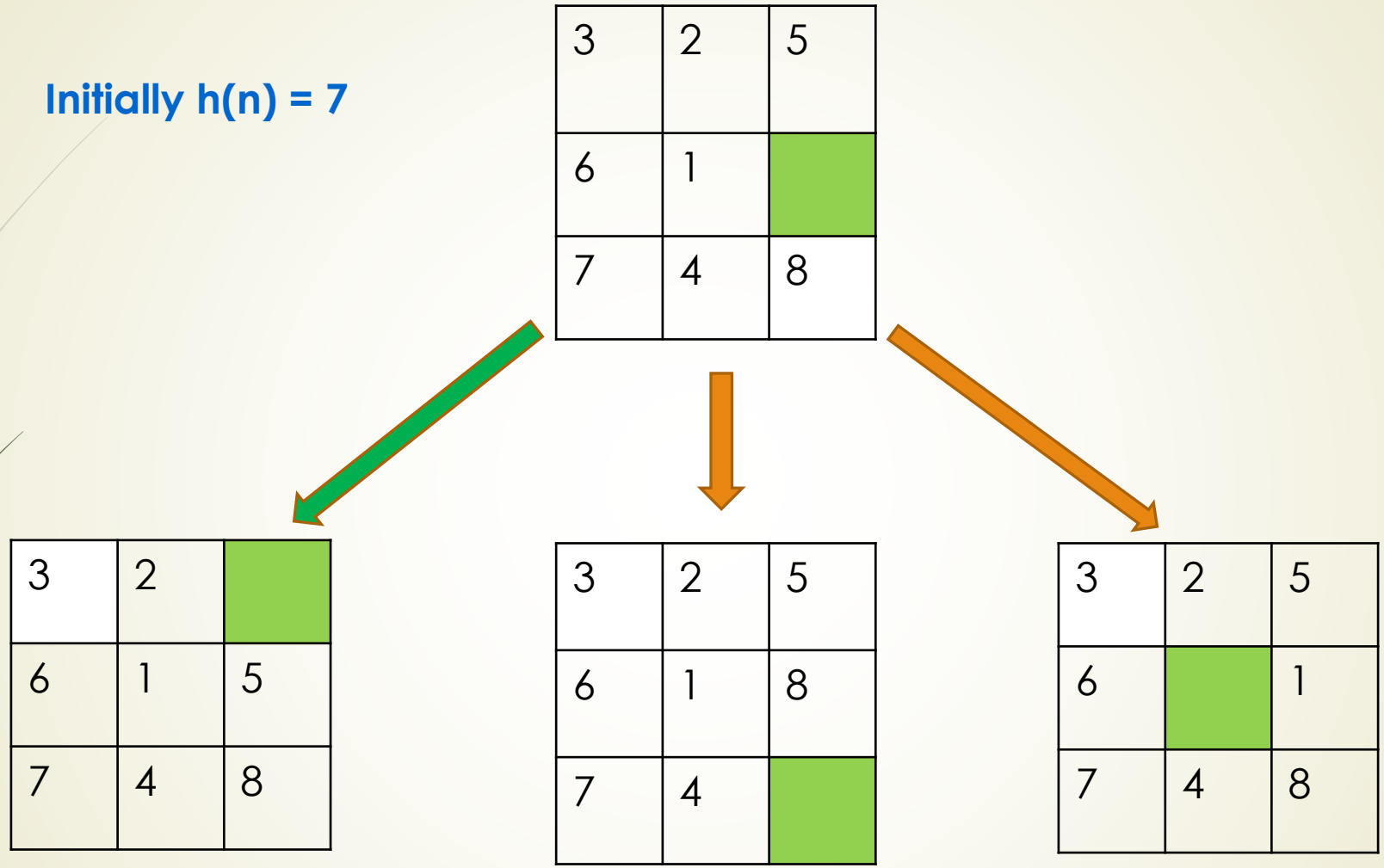
3	2	5
6	1	
7	4	8

Initial state

	1	2
3	4	5
6	7	8

Final state

Initially h(n) = 7



Move up ; h(n) = 6

Move down ; h(n) = 7

Move left ; h(n) = 7

Here the block will move up

Now h(n) = 6

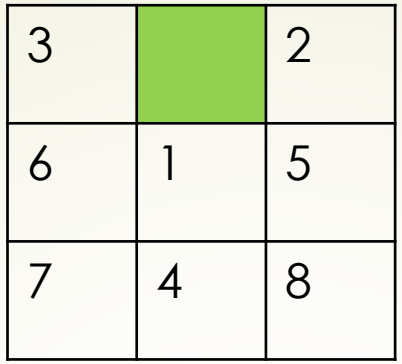
3	2	
6	1	5
7	4	8

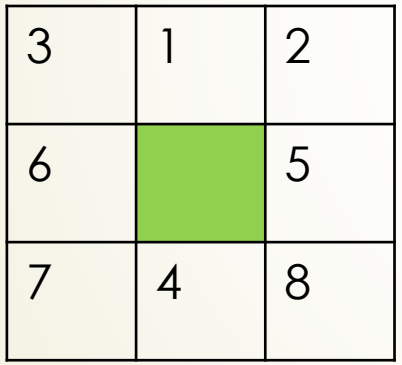
3		2
6	1	5
7	4	8

Move left ; h(n) = 5

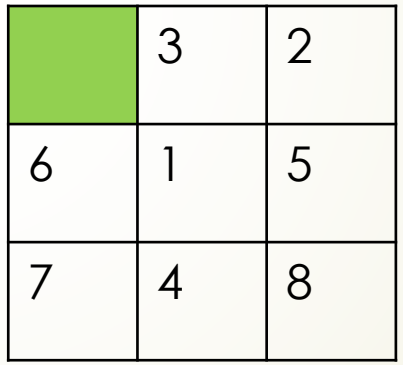
Here the block will move left

Now h(n) = 5





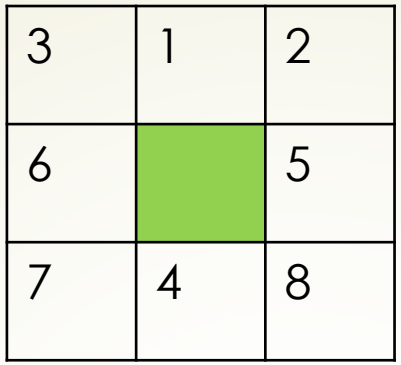
Move down; h(n) = 4

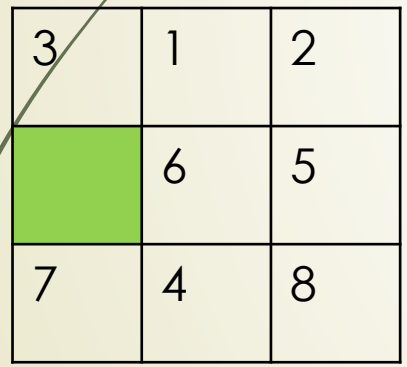


Move up ; h(n) = 5

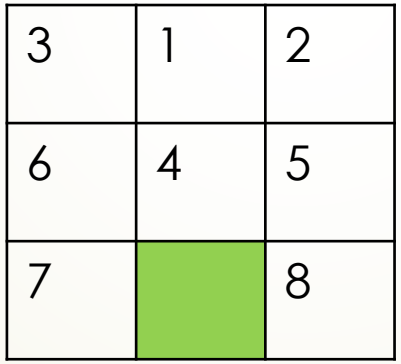
Here the block will move down

Now h(n) = 4

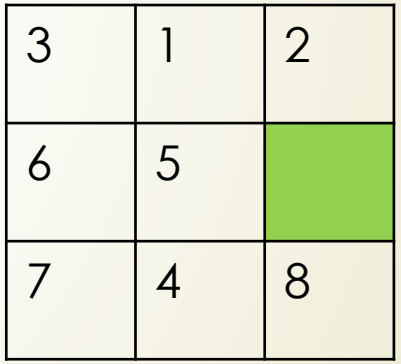




Move left ; h(n) = 4



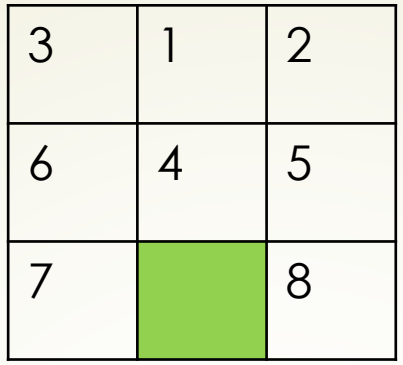
Move down ; h(n) = 3

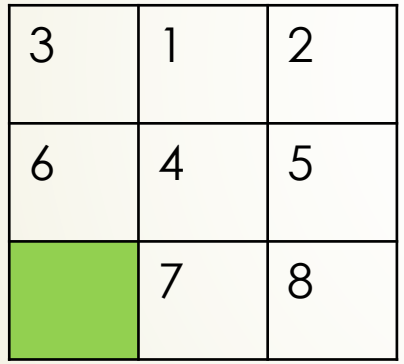


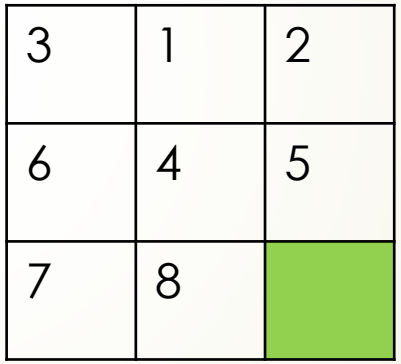
Move right ; h(n) = 5

Here the block will move down

Now h(n) = 3





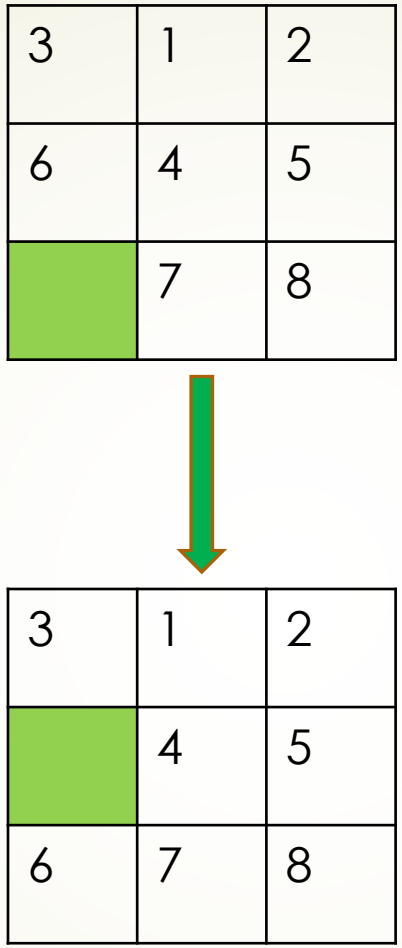


Move left ; h(n) =2

Move right ; h(n) = 4

Here the block will move left

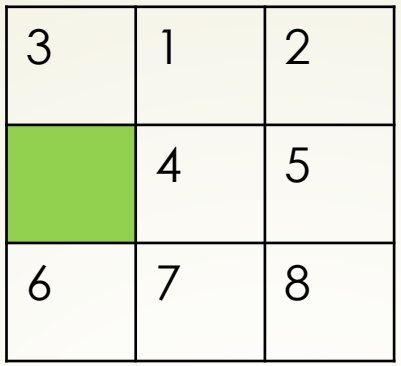
Now h(n) = 2

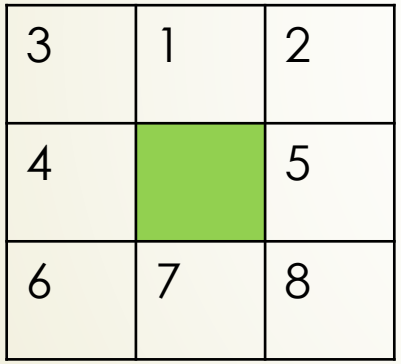


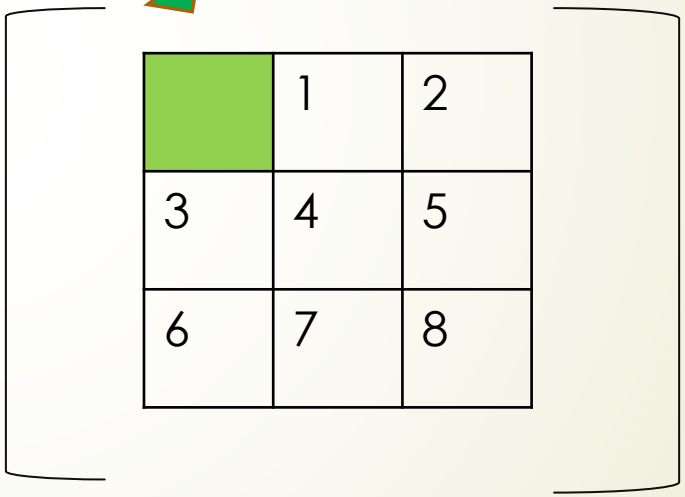
Move up ; h(n) = 1

Here the block will move up

Now h(n) = 1







Move right; h(n) =2

Move up ; h(n) = 0

This is the required final state. So here if the block moves up , the h(n) value becomes 0 and we get the result.

CONCLUSION

- We receive the optimal solution only for some of the specific inputs .
- We will not always get optimal solution.
- If the value of heuristic function does not decrease, it gets stuck in loops.

Thank You