Chapter 12: A* Pathfinding

Section 12.1: Introduction to A*

A* (A star) is a search algorithm that is used for finding path from one node to another. So it can be compared with Breadth First Search, or Dijkstra's algorithm, or Depth First Search, or Best First Search. A* algorithm is widely used in graph search for being better in efficiency and accuracy, where graph pre-processing is not an option.

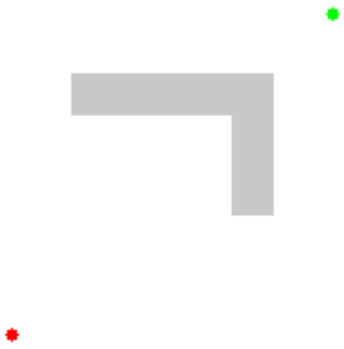
A* is a an specialization of Best First Search , in which the function of evaluation *f* is define in a particular way.

f(n) = g(n) + h(n) is the minimum cost since the initial node to the objectives conditioned to go thought node *n*.

g(n) is the minimum cost from the initial node to *n*.

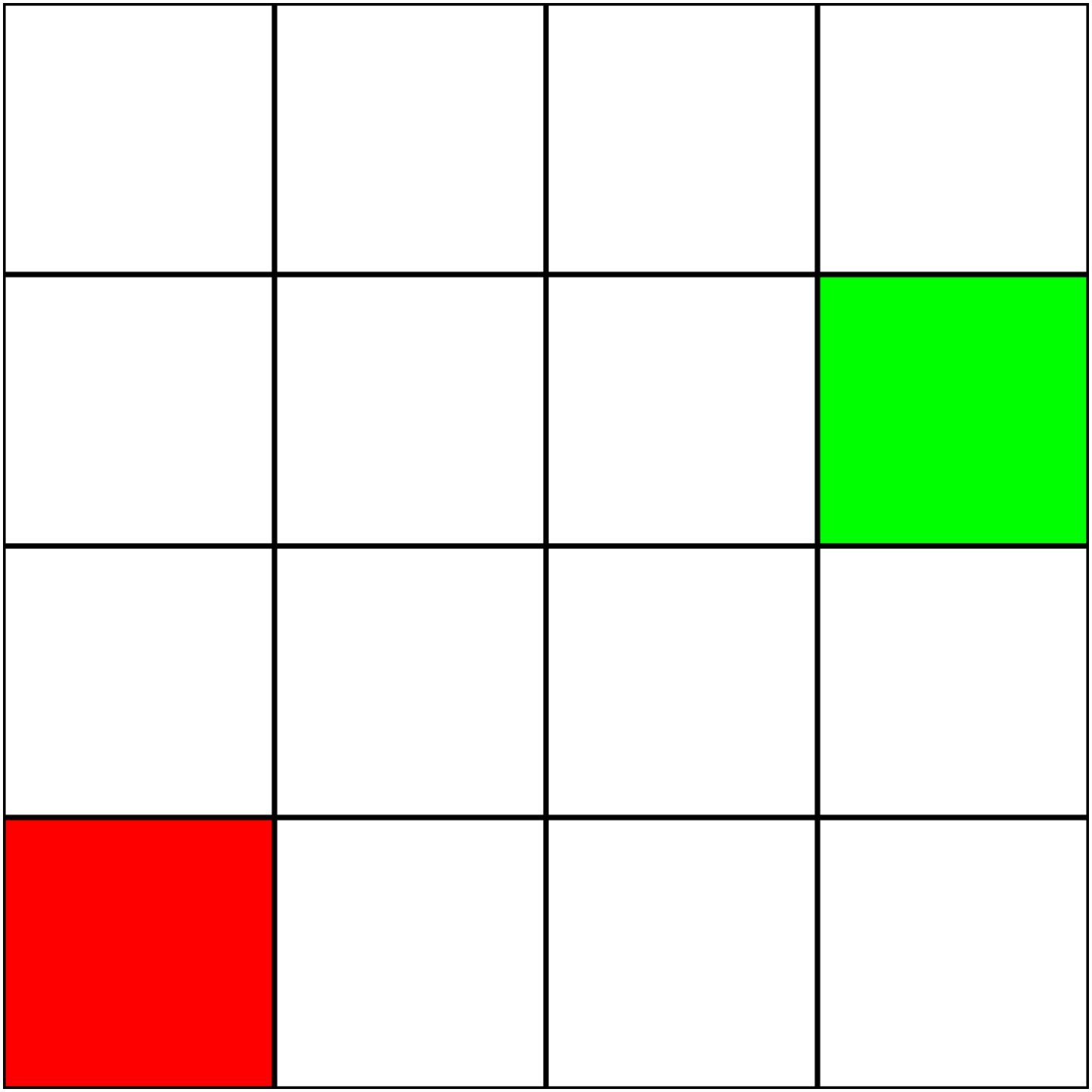
h(n) is the minimum cost from *n* to the closest objective to *n*

A* is an informed search algorithm and it always guarantees to find the smallest path (path with minimum cost) in the least possible time (if uses admissible heuristic). So it is both *complete* and *optimal*. The following animation demonstrates A* search-

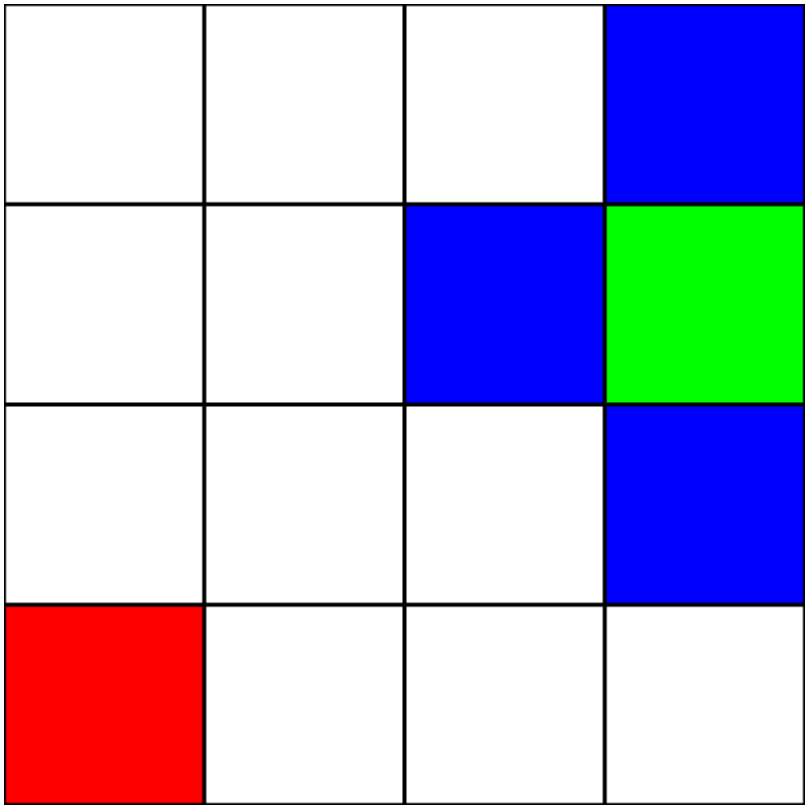


Section 12.2: A* Pathfinding through a maze with no obstacles

Let's say we have the following 4 by 4 grid:



Let's assume that this is a *maze*. There are no walls/obstacles, though. We only have a starting point (the green square), and an ending point (the red square). Let's also assume that in order to get from green to red, we cannot move diagonally. So, starting from the green square, let's see which squares we can move to, and highlight them in blue:



In order to choose which square to move to next, we need to take into account 2 heuristics:

1. The "g" value - This is how far away this node is from the green square.
2. The "h" value - This is how far away this node is from the red square.
3. The "f" value - This is the sum of the "g" value and the "h" value. This is the final number which tells us which node to move to.

In order to calculate these heuristics, this is the formula we will use: `distance = abs(from.x - to.x) + abs(from.y - to.y)`

This is known as the "Manhattan Distance" formula.

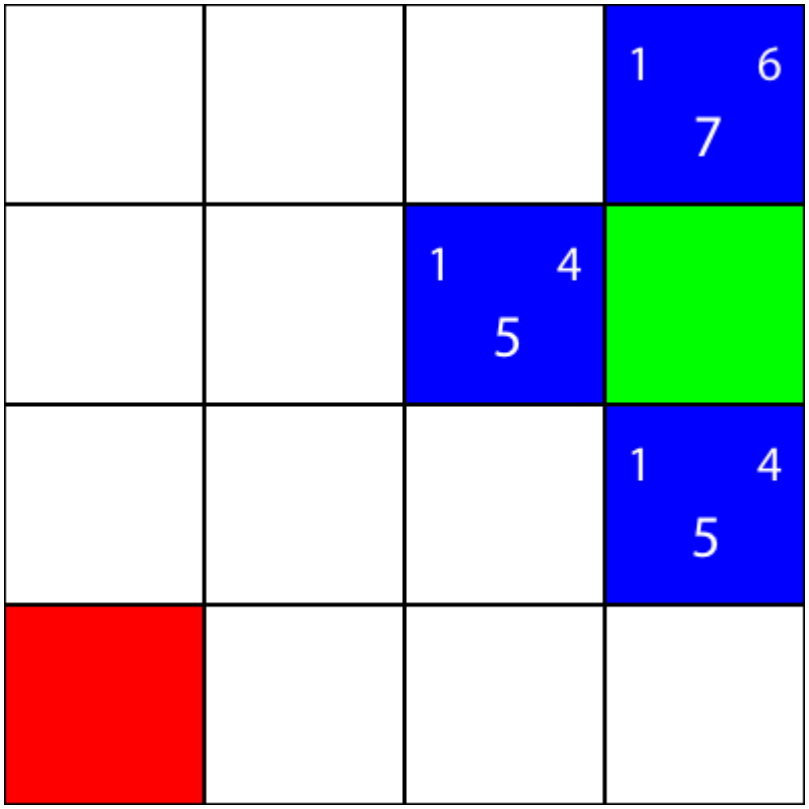
Let's calculate the "g" value for the blue square immediately to the left of the green square: `abs(3 - 2) + abs(2 - 2) = 1`

Great! We've got the value: 1. Now, let's try calculating the "h" value: `abs(2 - 0) + abs(2 - 0) = 4`

Perfect. Now, let's get the "f" value: `1 + 4 = 5`

So, the final value for this node is "5".

Let's do the same for all the other blue squares. The big number in the center of each square is the "f" value, while the number on the top left is the "g" value, and the number on the top right is the "h" value:



We've calculated the g, h, and f values for all of the blue nodes. Now, which do we pick?

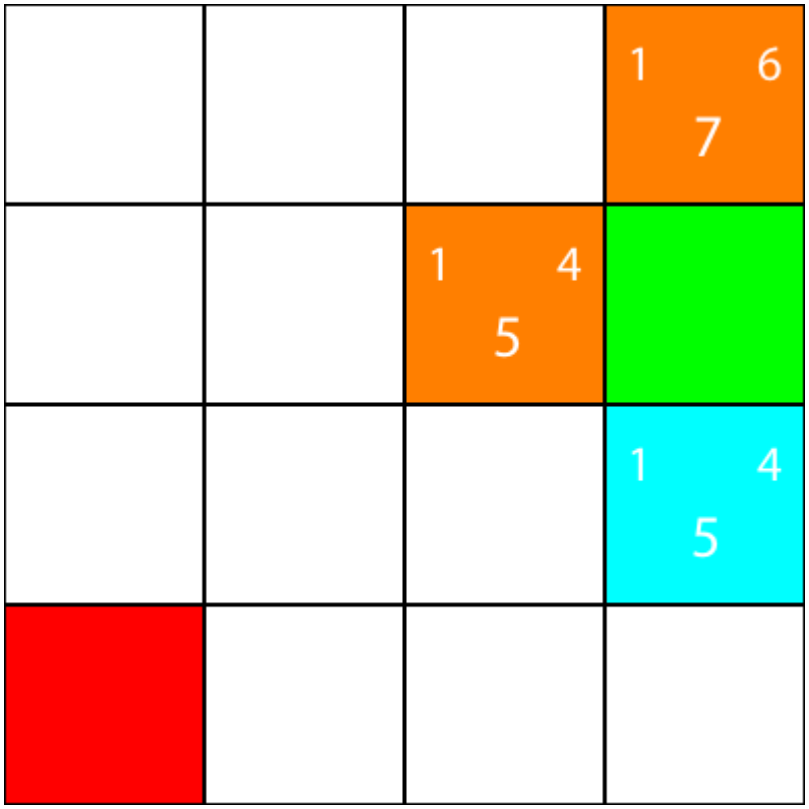
Whichever one has the lowest f value.

However, in this case, we have 2 nodes with the same f value, 5. How do we pick between them?

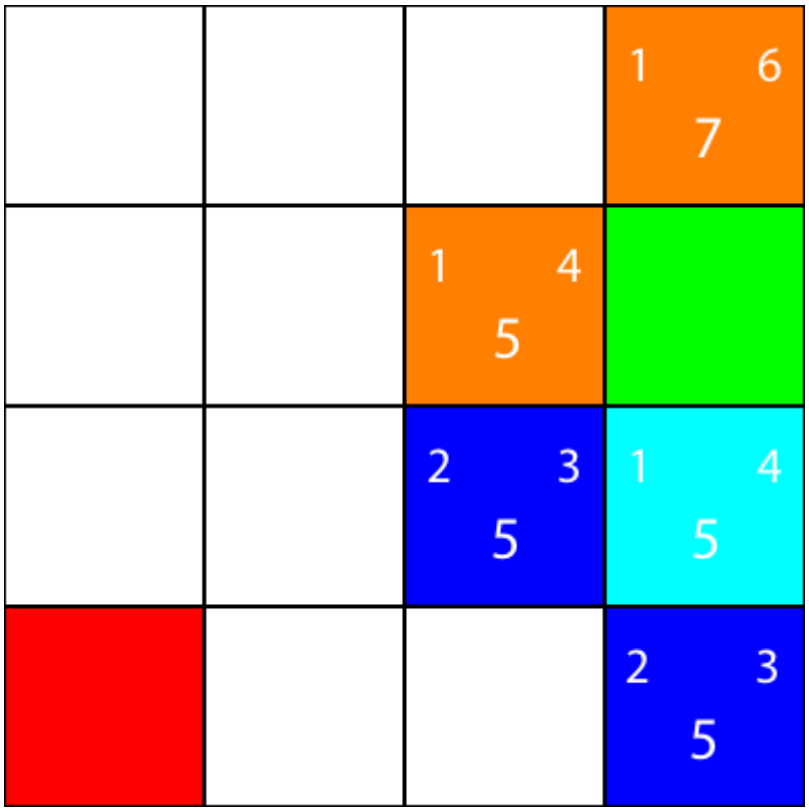
Simply, either choose one at random, or have a priority set. I usually prefer to have a priority like so: "Right > Up > Down > Left"

One of the nodes with the f value of 5 takes us in the "Down" direction, and the other takes us "Left". Since Down is at a higher priority than Left, we choose the square which takes us "Down".

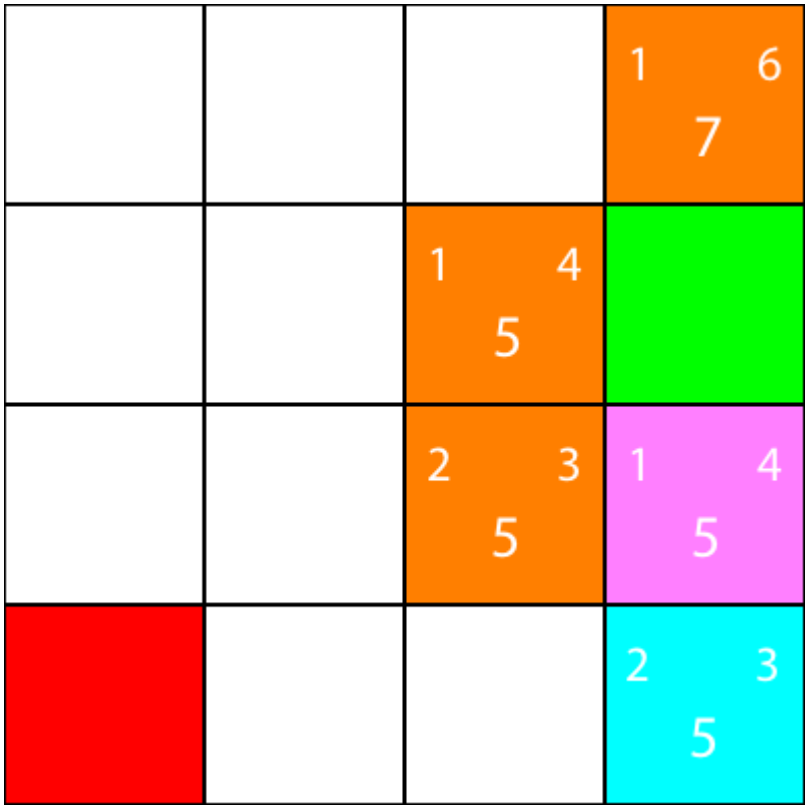
I now mark the nodes which we calculated the heuristics for, but did not move to, as orange, and the node which we chose as cyan:



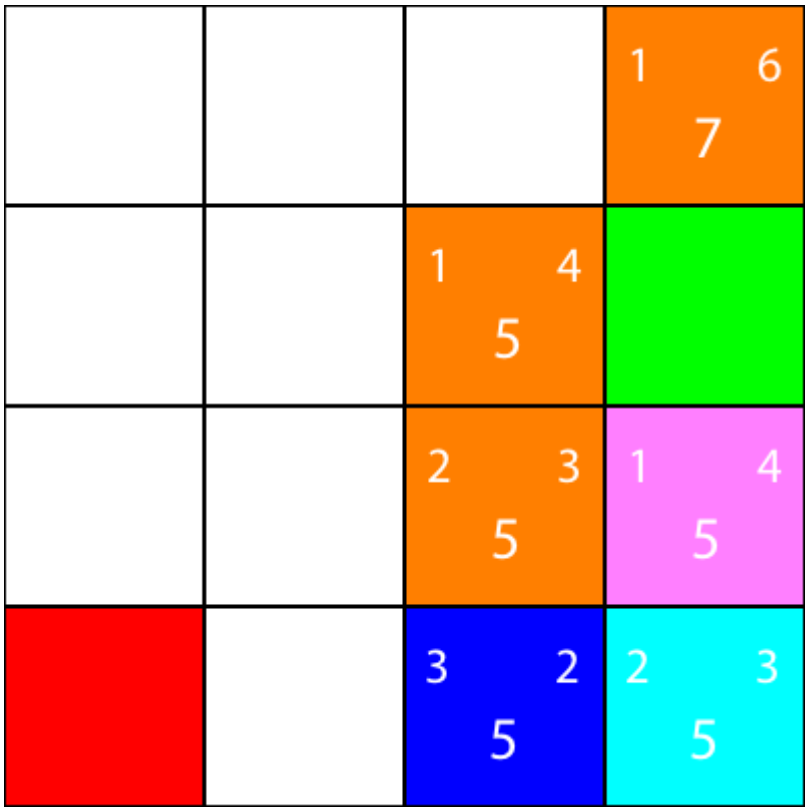
Alright, now let's calculate the same heuristics for the nodes around the cyan node:



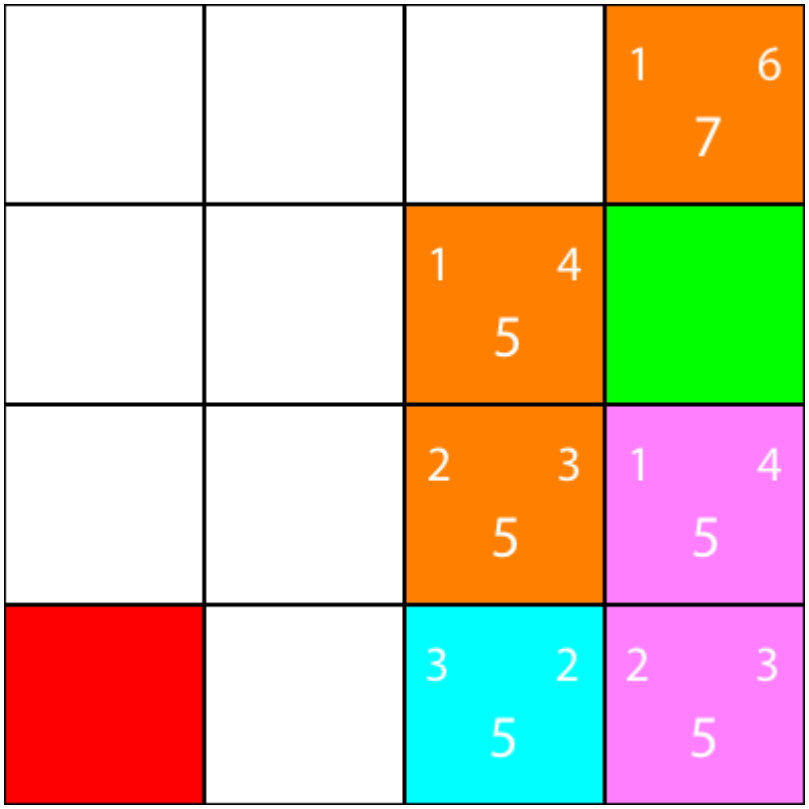
Again, we choose the node going down from the cyan node, as all the options have the same f value:



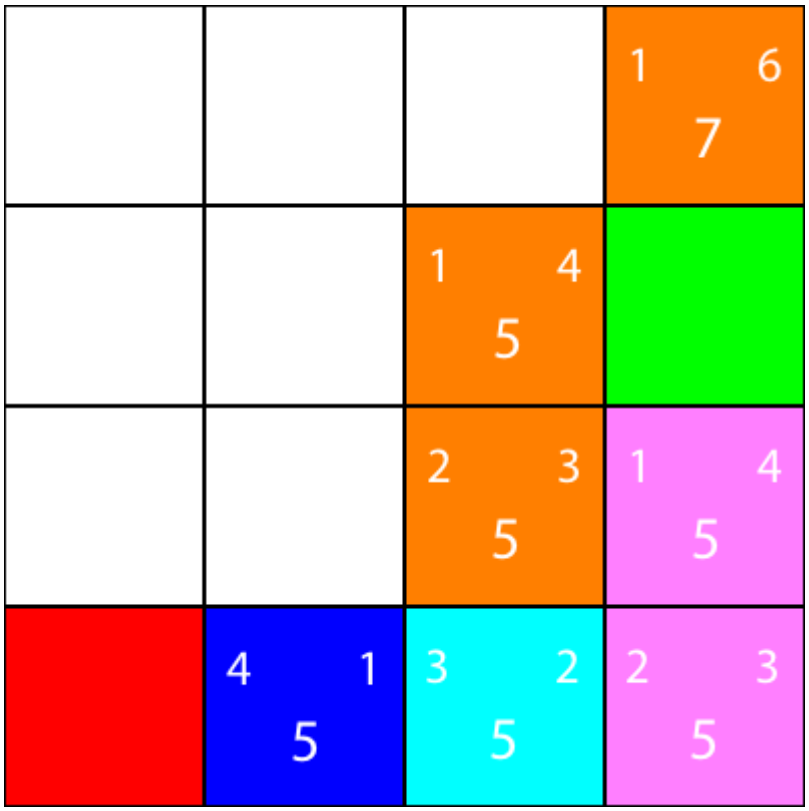
Let's calculate the heuristics for the only neighbour that the cyan node has:



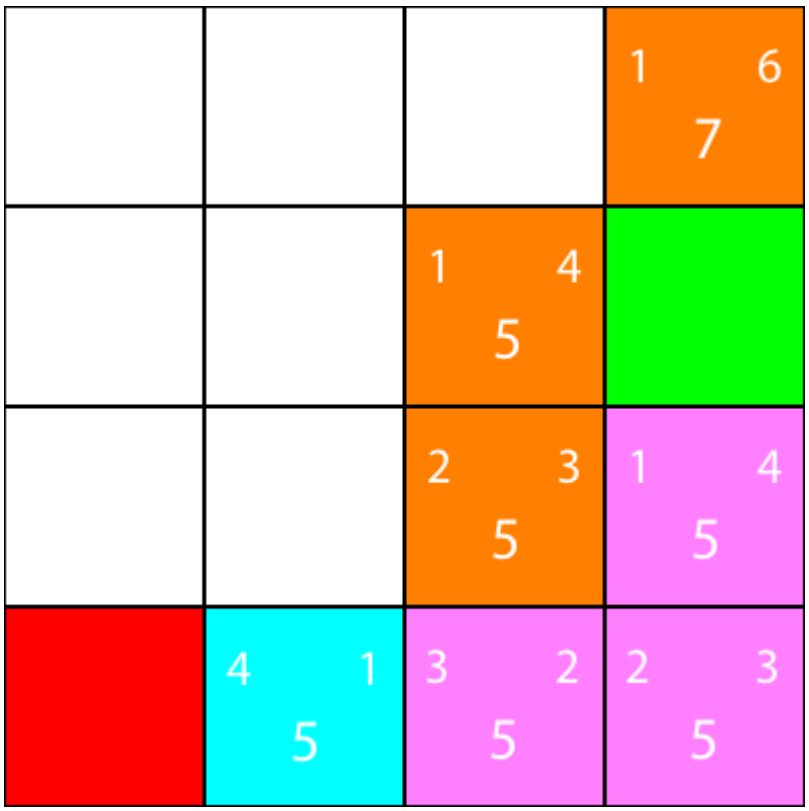
Alright, since we will follow the same pattern we have been following:



Once more, let's calculate the heuristics for the node's neighbour:



Let's move there:

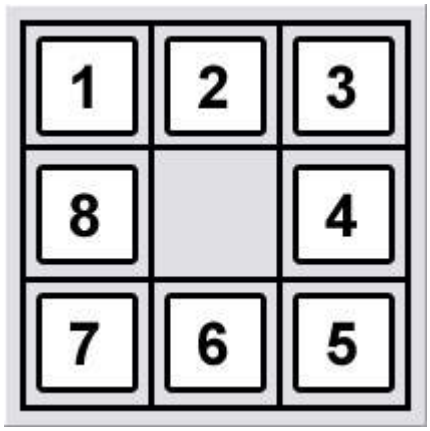


Finally, we can see that we have a *winning square* beside us, so we move there, and we are done.

Section 12.3: Solving 8-puzzle problem using A* algorithm

> **Problem definition**:

An 8 puzzle is a simple game consisting of a 3 x 3 grid (containing 9 squares). One of the squares is empty. The object is to move to squares around into different positions and having the numbers displayed in the "goal state".



Given an initial state of 8-puzzle game and a final state of to be reached, find the most cost-effective path to reach the final state from initial state.

> **Initial state**:

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

> **Final state**:

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

> **Heuristic to be assumed**:

Let us consider the Manhattan distance between the current and final state as the heuristic for this problem statement.

```
h(n) = | x - p | + | y - q |
where x and y are cell co-ordinates in the current state
      p and q are cell co-ordinates in the final state
```

> **Total cost function**:

So the total cost function `f(n)` is given by,

```
f(n) = g(n) + h(n), where g(n) is the cost required to reach the current state from given initial state
```

> **Solution to example problem**:

First we find the heuristic value required to reach the final state from initial state. The cost function, g(n) = 0, as we are in the initial state

```
h(n) = 8
```

The above value is obtained, as `1` in the current state is 1 horizontal distance away than the `1` in final state. Same goes for `2`, `5`, `6`. `_` is 2 horizontal distance away and 2 vertical distance away. So total value for `h(n)` is 1 + 1 + 1 + 1 + 2 + 2 = 8. Total cost function `f(n)` is equal to 8 + 0 = 8.

Now, the possible states that can be reached from initial state are found and it happens that we can either move `_` to right or downwards.

So states obtained after moving those moves are:

```
1 _ 3    4 1 3
4 2 5    _ 2 5
7 8 6    7 8 6
 (1)      (2)
```

Again the total cost function is computed for these states using the method described above and it turns out to be 6 and 7 respectively. We chose the state with minimum cost which is state (1). The next possible moves can be Left, Right or Down. We won't move Left as we were previously in that state. So, we can move Right or Down.

Again we find the states obtained from (1).

```
1 3 _    1 2 3
```

```
4 2 5     4 _ 5
7 8 6     7 8 6
 (3)       (4)
```

(3) leads to cost function equal to 6 and (4) leads to 4. Also, we will consider (2) obtained before which has cost function equal to 7. Choosing minimum from them leads to (4). Next possible moves can be Left or Right or Down. We get states:

```
1 2 3     1 2 3     1 2 3
_ 4 5     4 5 _     4 8 5
7 8 6     7 8 6     7 _ 6
 (5)       (6)       (7)
```

We get costs equal to 5, 2 and 4 for (5), (6) and (7) respectively. Also, we have previous states (3) and (2) with 6 and 7 respectively. We chose minimum cost state which is (6). Next possible moves are Up, and Down and clearly Down will lead us to final state leading to heuristic function value equal to 0.