# Assignment One

## Algorithm walk through

| **Start State** | | |
|---|---|---|
| 1 | 2 | 3 |
| | 4 | 6 |
| 7 | 5 | 8 |

| **Goal State** | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

$x = g(n) + h(n)$

$g(n)$ = distance from the current position until the last position

$h(n)$ = numbers that are not in the correct order according to the goal state

| **Start State** | | |
|---|---|---|
| 1 | 2 | 3 |
| | 4 | 6 |
| 7 | 5 | 8 |

$g(n) = 0$

$h(n) = 3$

Result: $f(n) = 3$

## G = 1

| **Move up** | | |
|---|---|---|
| | 2 | 3 |
| 1 | 4 | 6 |
| 7 | 5 | 8 |

$g(n) = 1$

$h(n) = 4$

Result: $f(n) = 5$

| **Move Down** | | |
|---|---|---|
| 1 | 2 | 3 |
| 7 | 4 | 6 |
| | 5 | 8 |

$g(n) = 1$

$h(n) = 4$

Result: $f(n) = 5$

| **Move Right** | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | | 6 |
| 7 | 5 | 8 |

$g(n) = 1$

$h(n) = 2$

Result: $f(n) = 3$

## G=2

| **Move Left** | | |
|---|---|---|
| 1 | 2 | 3 |
| | 4 | 6 |
| 7 | 5 | 8 |

$g(n) = 2$

$h(n) = 3$

Result: $f(n) = 5$

| **Move Down** | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | | 8 |

$g(n) = 2$

$h(n) = 1$

Result: $f(n) = 3$

| **Move Right** | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 6 | |
| 7 | 5 | 8 |

$g(n) = 2$

$h(n) = 3$

Result: $f(n) = 5$

| **Move Up** | | |
|---|---|---|
| 1 | | 3 |
| 4 | 2 | 6 |
| 7 | 5 | 8 |

$g(n) = 2$

$h(n) = 3$

Result: $f(n) = 5$

## G = 3

| **Move Left** | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| | 7 | 8 |

## Winner!!!

| **Move Right** | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

$g(n) = 3$

$h(n) = 0$

Result: $f(n) = 3$

| Move Down | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | | 6 |
| 7 | 5 | 8 |

## Output After Algorithm

## Code For Project

```
import react from "react";
import { useState, useEffect } from "react";
import reactdom from "react-dom";
import {
  ChakraProvider,
  Center,
  Heading,
  Text,
  Link,
  Button,
  Badge,
  HStack,
  VStack,
} from "@chakra-ui/react";

function App() {
  const board = [
    [1, 7, 8],
```

```
    [3, 0, 4],
    [6, 2, 5],
  ];
  const goalstate = [
    [0, 3, 6],
    [1, 4, 7],
    [2, 5, 8],
  ];

  const [top, settop] = useState(false);
  const [right, setright] = useState(false);
  const [left, setleft] = useState(false);
  const [bottom, setbottom] = useState(false);
  const [goal, setgoal] = useState(false);
  const [zerodir, setzerodir] = useState("nowhere");
  const [boardstate, setboardstate] = useState(board);
  const [inputindex, setinputindex] = useState(0);
  const [inputsubindex, setinputsubindex] = useState(0);
  const [corrects, setcorrects] = useState([]);
  const [g, setg] = useState(0);
  const [h, seth] = useState([]);
  const [pressed, setpressed] = useState(false);

  function AI() {
    var zeroindex = 0;
    var zerosubindex = 0;

    for (let i = 0; i < boardstate.length; i++) {
      for (let t = 0; t < boardstate.length; t++) {
        if (boardstate[i][t] === 0) {
          zeroindex = i;
          zerosubindex = t;
        }
      }
    }
    console.log("Zero was at:" + zeroindex + ", " + zerosubindex);
    if (zeroindex === 0 && zerosubindex === 0) {
      console.log("I am top left");
      var temp0 = [[], [], []];
      var temp1 = [[], [], []];
    }
    if (zeroindex === 0 && zerosubindex === 1) {
```

```
      console.log("I am middle left");
      var temp0 = [[], [], []];
      var temp1 = [[], [], []];
      var temp2 = [[], [], []];
    }
    if (zeroindex === 0 && zerosubindex === 2) {
      console.log("I am bottom left");
    }
    if (zeroindex === 1 && zerosubindex === 0) {
      console.log("I am top middle");
    }
    if (zeroindex === 1 && zerosubindex === 1) {
      console.log("I am middle middle");
      var temp0 = [[], [], []];
      var temp1 = [[], [], []];
      var temp2 = [[], [], []];
      var temp3 = [[], [], []];
    }
    if (zeroindex === 1 && zerosubindex === 2) {
      console.log("I am bottom middle");

    }
    if (zeroindex === 2 && zerosubindex === 0) {
      console.log("I am top right");
    }
    if (zeroindex === 2 && zerosubindex === 1) {
      console.log("I am middle right");
    }
    if (zeroindex === 2 && zerosubindex === 2) {
      console.log("I am bottom right");
    }
  }

  useEffect(() => {
    if (boardstate === goalstate) {
      setgoal(true);
    }
    for (let i = 0; i < boardstate.length; i++) {
      for (let t = 0; t < boardstate.length; t++) {
        if (boardstate[i][t] === goalstate[i][t]) {
          seth(h.splice(h.length, 0, boardstate[i][t]));
        } else {
```

```
        }
      }
    }
    setcorrects(h.flat());
  }, [pressed]);
  function handleClick(index, sindex) {
    AI();
    setpressed(!pressed);
    var zeroindex = 0;
    var zerosindex = 0;
    var value = boardstate[index][sindex];

    for (let i = 0; i < boardstate.length; i++) {
      for (let t = 0; t < boardstate.length; t++) {
        if (boardstate[i][t] === 0) {
          zeroindex = i;
          zerosindex = t;
        }
      }
    }

    {
      /* console.log("Button Pressed:" + index + ", " + sindex);
    console.log("Board Index 0: " + boardstate[0]);
    console.log("Board Index 1: " + boardstate[1]);
    console.log("Board Index 2: " + boardstate[2]);
    console.log("Zero Index was : " + zeroindex + " " + zerosindex);
*/
    }

    if (sindex < board.length - 1) {
      if (boardstate[index][sindex + 1] === boardstate[zeroindex]
[zerosindex]) {
        var temp = [...boardstate[index]];
        temp.splice(sindex, 1, 0);
        temp.splice(sindex + 1, 1, value);
        if (index === 0) {
          setboardstate([temp, boardstate[1], boardstate[2]]);
        }
        if (index === 1) {
          setboardstate([boardstate[0], temp, boardstate[2]]);
        }
```

```
        if (index === 2) {
          setboardstate([boardstate[0], boardstate[1], temp]);
        }
      }
    }
    if (sindex > 0) {
      if (boardstate[index][sindex - 1] === boardstate[zeroindex]
[zerosindex]) {
        var temp = [...boardstate[index]];
        temp.splice(sindex, 1, 0);
        temp.splice(sindex - 1, 1, value);
        if (index === 0) {
          setboardstate([temp, boardstate[1], boardstate[2]]);
        }
        if (index === 1) {
          setboardstate([boardstate[0], temp, boardstate[2]]);
        }
        if (index === 2) {
          setboardstate([boardstate[0], boardstate[1], temp]);
        }
      }
    }

    if (index < board.length - 1) {
      if (boardstate[index + 1][sindex] === boardstate[zeroindex]
[zerosindex]) {
        if (index === 1) {
          var temprowzero = boardstate[index].splice(sindex, 1, 0);
          var temprow = boardstate[zeroindex].splice(zerosindex, 1,
value);
          var temp1 = boardstate.splice(index, 1,
[temprowzero]).flat();
          var temp0 = [...boardstate[0]];
          var temp2 = boardstate.splice(zeroindex, 1,
[temprow]).flat();
          setboardstate([temp0, temp1, temp2]);
        } else if (index === 0) {
          var temprowzero = boardstate[index].splice(sindex, 1, 0);
          var temp0 = boardstate.splice(index, 1,
[temprowzero]).flat();
          var temprow = boardstate[zeroindex].splice(zerosindex, 1,
value);
```

```
          var temp2 = [...boardstate[2]];
          var temp1 = boardstate.splice(zeroindex, 1,
[temprow]).flat();

          setboardstate([temp0, temp1, temp2]);
        }
      }
    }
    if (index > 0) {
      if (board[index - 1][sindex] === board[zeroindex][zerosindex])
{
        console.log("swap left zero ");
        if (index === 2) {
          var temprowzero = boardstate[index].splice(sindex, 1, 0);
          var temprow = boardstate[zeroindex].splice(zerosindex, 1,
value);
          var temp1 = [...boardstate[1]];
          var temp0 = [...boardstate[0]];
          var temp2 = boardstate.splice(index, 1, [temprow]).flat();

          setboardstate([temp0, temp1, temp2]);
        } else if (index === 1) {
          var temprowzero = boardstate[index].splice(sindex, 1, 0);
          var temprow = boardstate[zeroindex].splice(zerosindex, 1,
value);
          var temp2 = [...boardstate[2]];
          var temp0 = boardstate.splice(zeroindex, 1,
[temprowzero]).flat();
          var temp1 = boardstate.splice(index, 1, [temprow]).flat();

          setboardstate([temp0, temp1, temp2]);
        }
      }
    }
  }

  return (
    <Center p="20">
      <Text>H is {corrects.length}</Text>
      <HStack>
        {boardstate.map((items, index) => {
          return (
```

```
            <VStack>
              {items.map((sitem, sindex) => {
                return (
                  <Button
                    size="lg"
                    colorScheme={corrects.includes(sitem) ? "red" :
"blue"}
                    opacity={sitem === 0 ? "0" : "1"}
                    onClick={() => {
                      handleClick(index, sindex);
                    }}
                  >
                    {sitem}
                  </Button>
                );
              })}
            </VStack>
          );
        })}
      </HStack>
    </Center>
  );
}

reactdom.render(
  <ChakraProvider>
    <App />
  </ChakraProvider>,
  document.getElementById("root")
);
```