8-Puzzle BFS Algorithm

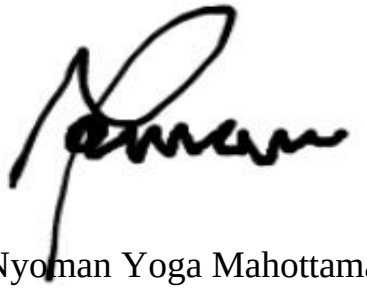
DAA D – Quiz 2

By :

- Naulia Adam Kuncoro (05111740000155)
- I Nyoman Yoga Mahottama (05111740000159)
- Faishal Abiyyudzakir (05111740000192)

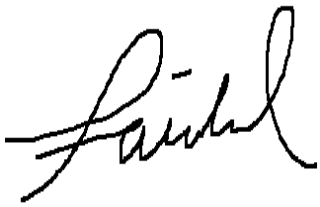
By the name of Allah (God) Almighty, herewith we pledge and truly declare that we have solved quiz 2 by ourselves, didn't do any cheating by any means, didn't do any plagiarism, and didn't accept anybody's help by any means. We are going to accept all of the consequences by any means if it has proven that we have been done any cheating and/or plagiarism

Surabaya, 25 March 2020



[I Nyoman Yoga Mahottama]
[05111740000159]

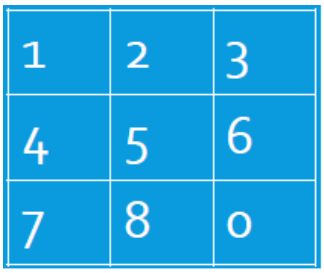
[Naulia Adam Kuncoro]
[05111740000155]



[Faishal A]
[05111740000192]

- 8-PUZZLE

The 8-Puzzle is a puzzle which played on a 3-by-3 grid with 8 square blocks labeled 1 through 8 and a blank square. The goal is to rearrange the blocks so that they are in goal state. You are permitted to slide blocks horizontally or vertically into the blank square.



- SOURCE CODE AND ANALYSIS:

```
int main()
{
    int i,j, x0,y0;
    int puzzle[N][N], goal[N][N];
    printf("Input initial state : \n");
    int input;
    for(i=0;i<N;i++)
        for(j=0;j<N;j++){
            scanf("%d",&puzzle[i][j]);
        }
    printf("Input goal state : \n");
    for(i=0;i<N;i++)
        for(j=0;j<N;j++){
            scanf("%d",&goal[i][j]);
        }
```

First of all, In Main Function we must input initial state and goal state that we want in this 8 puzzle. After we search the zero coordinates in the puzzle 8 and we store it in the x0 angy0 variable with this code :

```
for(i=0;i<N;i++)
    for(j=0;j<N;j++){
        if(puzzle[i][j]==0){
            x0 = i; y0=j;
            break;
        }
    }
printf("wait for solution...\n");
solve(puzzle,x0,y0,goal);
```

Then we call Solve() function. Inside of the solve function:

```
void solve(int initial[N][N], int x, int y,
           int final[N][N])
{
    queue<Node*> pq;

    Node* root = newNode(initial, x, y, x, y, 0, NULL);
    root->cost = calculateCost(initial, final);
    pq.push(root);
```

First, make new queue called pq, then makes another new nodes called root that store initial matrix, and calculate the cost. Cost is total of the misplaced tiles inside of the puzzle8 using function calculate cost. Then push the root inside the queue.

After pushed, it went inside the looping section:

```
while (!pq.empty())
{
    Node* min = pq.front();
    pq.pop();
    if (min->cost == 0)
    {
        printf("Solution found : \n");
        printPath(min);
        return;
    }
```

While the queue isn't empty, make new variable called min that store the most front data inside the queue, then pop the queue. After that, check does the cost is zero, that mean 8 puzzle already reach the goal state. If it hasn't, so continue the into:

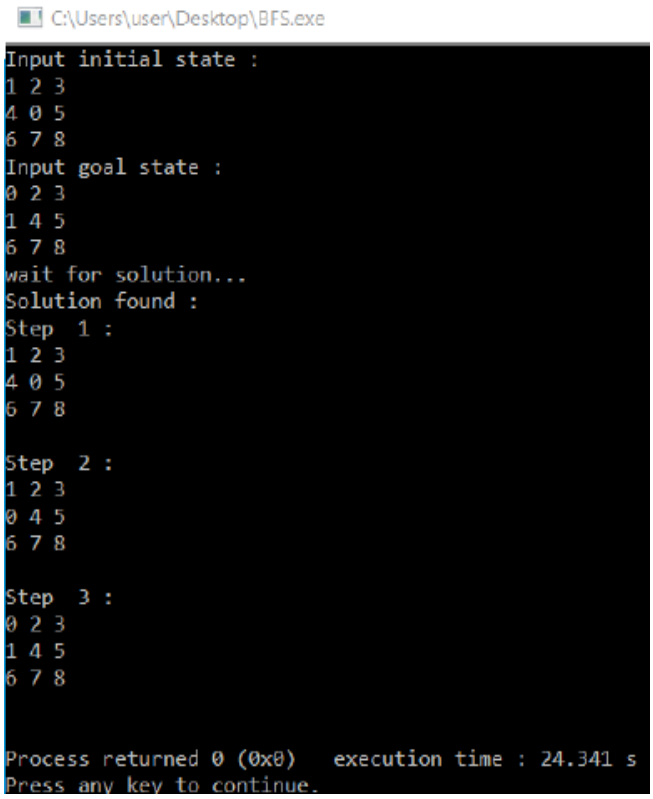
```
for (int i = 0; i < 4; i++)
{
    if (isSafe(min->x + row[i], min->y + col[i]))
    {
        Node* child = newNode(min->mat, min->x,
                              min->y, min->x + row[i],
                              min->y + col[i],
                              min->level + 1, min);
        child->cost = calculateCost(child->mat, final);
        h
        if(min->parent==NULL)
        pq.push(child);

        else{
            if(child->x!=min->parent->x && child->y!=min->parent->y)
            pq.push(child);
        }

    }
}
```

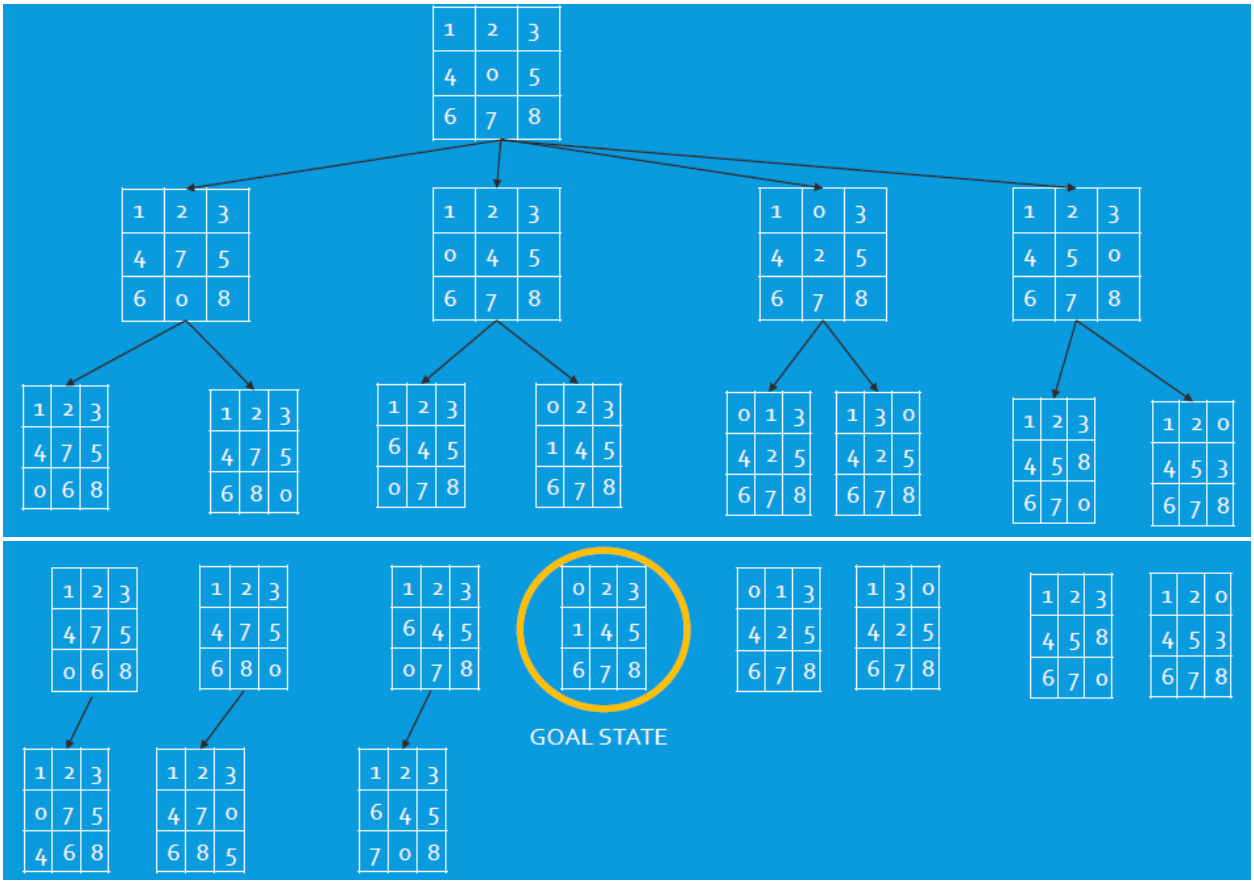
Expand The nodes using function newNodeswith priority movement down, left, up, and right as a achilndconnecting to the parent. Then push the child inside the queue.

- EXAMPLE OF OUTPUT:



- REPRESENTATION OF STATE :

- REPRESENTATION OF BFS EXAMPLE :



Github link :

https://github.com/yogamahottama/8Puzzle-DAA-D