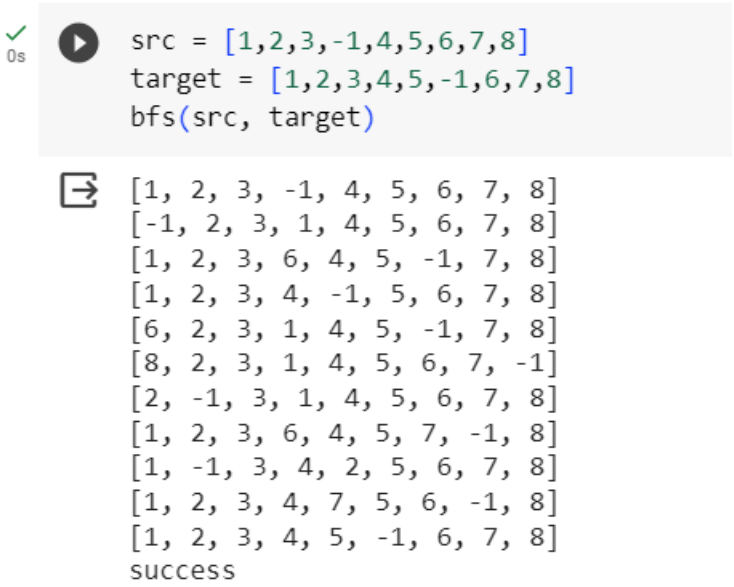
8-Puzzle

```
import numpy as np
import pandas as pd
import os

def bfs(src,target):
    queue = []
    queue.append(src)

    exp = []

    while len(queue) > 0:
        source = queue.pop(0)
        exp.append(source)

        print(source)

        if source==target:
            print("success")
            return

        poss_moves_to_do = []
        poss_moves_to_do = possible_moves(source,exp)

        for move in poss_moves_to_do:

            if move not in exp and move not in queue:
                queue.append(move)
def possible_moves(state,visited_states):
```

```
    b = state.index(-1)

    #directions array
    d = []


    if b not in [-1,1,2]:
        d.append('u')
    if b not in [6,7,8]:
        d.append('d')
    if b not in [-1,3,6]:
        d.append('l')
    if b not in [2,5,8]:
        d.append('r')


    pos_moves_it_can = []



    for i in d:
        pos_moves_it_can.append(gen(state,i,b))

    return [move_it_can for move_it_can in pos_moves_it_can if move_it_can not in
visited_states]
def gen(state, m, b):
    temp = state.copy()

    if m=='d':
        temp[b+3],temp[b] = temp[b],temp[b+3]
```

```
    if m=='u':
        temp[b-3],temp[b] = temp[b],temp[b-3]

    if m=='l':
        temp[b-1],temp[b] = temp[b],temp[b-1]

    if m=='r':
        temp[b+1],temp[b] = temp[b],temp[b+1]



    return temp
```

OUTPUT



```
src = [1,2,3,-1,4,5,6,7,8]
target = [1,2,3,4,5,-1,6,7,8]
bfs(src, target)
```

```
[1, 2, 3, -1, 4, 5, 6, 7, 8]
[-1, 2, 3, 1, 4, 5, 6, 7, 8]
[1, 2, 3, 6, 4, 5, -1, 7, 8]
[1, 2, 3, 4, -1, 5, 6, 7, 8]
[6, 2, 3, 1, 4, 5, -1, 7, 8]
[8, 2, 3, 1, 4, 5, 6, 7, -1]
[2, -1, 3, 1, 4, 5, 6, 7, 8]
[1, 2, 3, 6, 4, 5, 7, -1, 8]
[1, -1, 3, 4, 2, 5, 6, 7, 8]
[1, 2, 3, 4, 7, 5, 6, -1, 8]
[1, 2, 3, 4, 5, -1, 6, 7, 8]
success
```

0s

```
src = [2,-1,3,1,8,4,7,6,5]
target = [1,2,3,8,-1,4,7,6,5]
bfs(src, target)
```

```
[2, -1, 3, 1, 8, 4, 7, 6, 5]
[2, 8, 3, 1, -1, 4, 7, 6, 5]
[-1, 2, 3, 1, 8, 4, 7, 6, 5]
[2, 3, -1, 1, 8, 4, 7, 6, 5]
[2, 8, 3, 1, 6, 4, 7, -1, 5]
[2, 8, 3, -1, 1, 4, 7, 6, 5]
[2, 8, 3, 1, 4, -1, 7, 6, 5]
[7, 2, 3, 1, 8, 4, -1, 6, 5]
[1, 2, 3, -1, 8, 4, 7, 6, 5]
[5, 2, 3, 1, 8, 4, 7, 6, -1]
[2, 3, 4, 1, 8, -1, 7, 6, 5]
[2, 8, 3, 1, 6, 4, -1, 7, 5]
[2, 8, 3, 1, 6, 4, 7, 5, -1]
[-1, 8, 3, 2, 1, 4, 7, 6, 5]
[2, 8, 3, 7, 1, 4, -1, 6, 5]
[2, 8, -1, 1, 4, 3, 7, 6, 5]
[2, 8, 3, 1, 4, 5, 7, 6, -1]
[7, 2, 3, -1, 8, 4, 1, 6, 5]
[7, 2, 3, 1, 8, 4, 6, -1, 5]
[1, 2, 3, 7, 8, 4, -1, 6, 5]
[1, 2, 3, 8, -1, 4, 7, 6, 5]
success
```