

8 Puzzles and 8 Queens

Using DFS and BFS

Kelompok

Farrela Ranku Mahhisa 5025211129

Faizah Nurdianti Maghfirah 5025211134

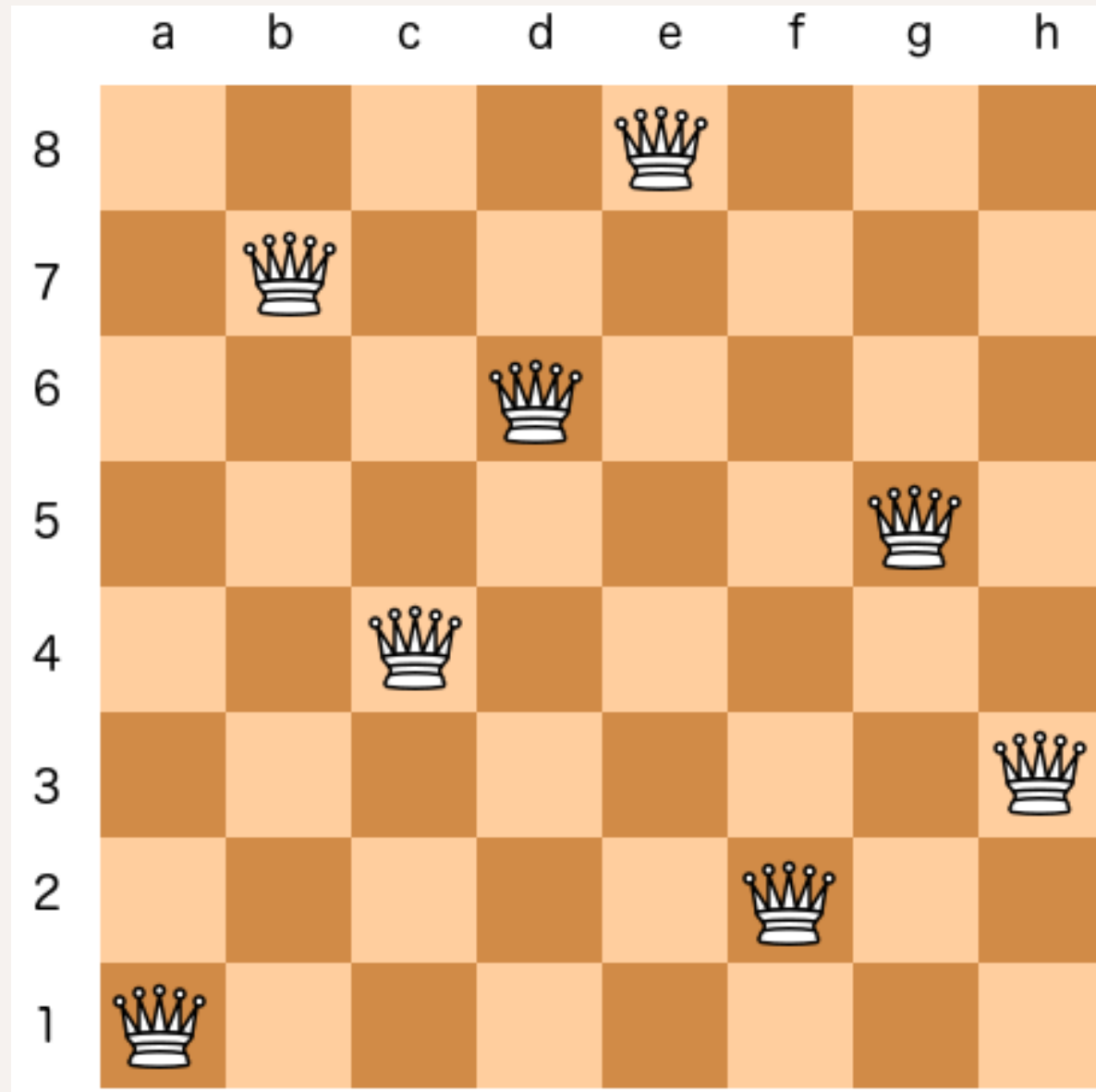
Shafa Nabilah Hanin 5025211222

1	2	3
4	5	6
7	8	

8 Puzzle

Masalah 8 puzzle adalah sebuah game puzzle di mana kita harus memindahkan ubin-ubin yang ada di dalam sebuah kotak 3x3 sehingga urutannya sama dengan urutan yang diinginkan, sambil membiarkan satu tempat kosong untuk memudahkan pergerakan. Kita bisa memindahkan ubin-ubin tersebut dengan cara menggesernya, tapi kita hanya bisa menggeser ubin yang bersebelahan dengan tempat kosong. Masalah ini sering digunakan sebagai contoh masalah pencarian langkah-langkah dalam kecerdasan buatan.

8 Queens



Masalah 8 queens adalah sebuah permasalahan kombinatorial di mana kita harus menempatkan 8 ratu pada papan catur berukuran 8x8 sedemikian sehingga tidak ada satu ratupun yang dapat menyerang ratu lainnya. Artinya, tidak ada dua ratu yang berada pada satu baris, kolom, atau diagonal yang sama. Masalah ini sering dijadikan sebagai contoh masalah dalam ilmu komputer dan matematika, karena sifat kombinatorialnya yang rumit dan memiliki banyak aplikasi dalam bidang-bidang seperti optimasi, pengambilan keputusan, dan teori graf.



BFS

Menggunakan queue

Lebih banyak memori yang digunakan


Kurang cocok digunakan untuk decision making trees dalam game / puzzle

DFS

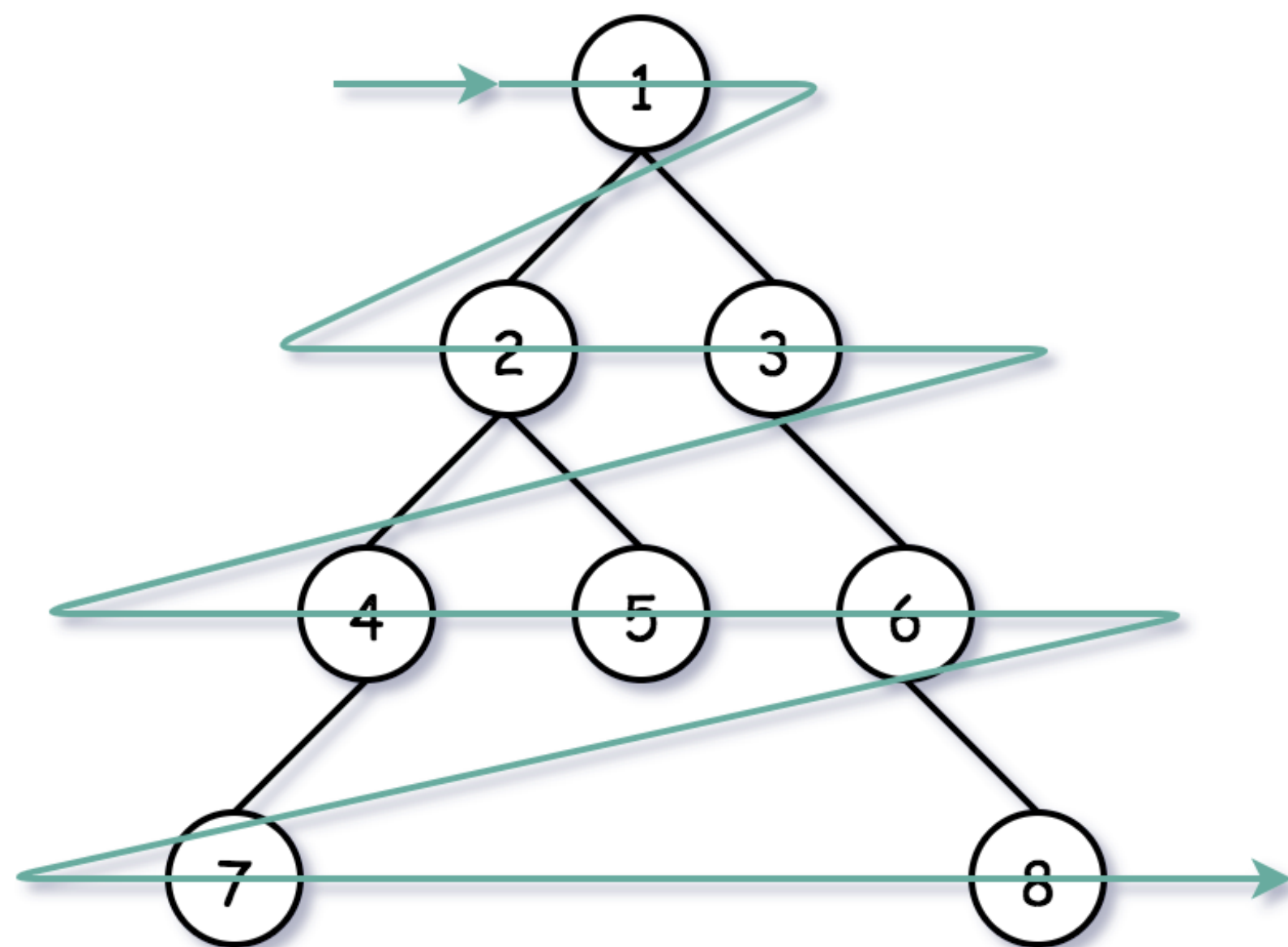
Menggunakan stack

Lebih sedikit memori yang digunakan

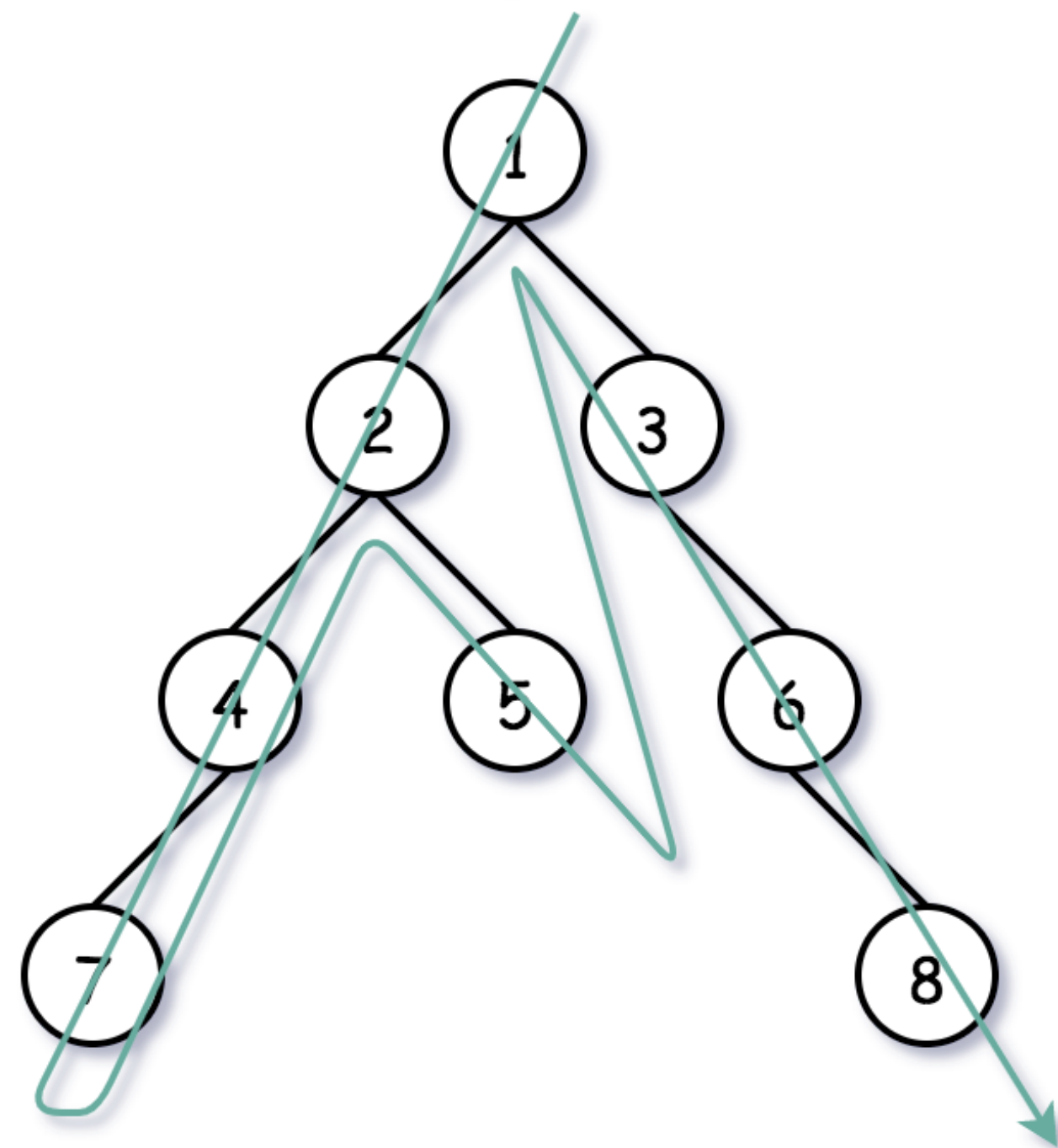
Cocok digunakan untuk decision making trees dalam game / puzzle



BFS



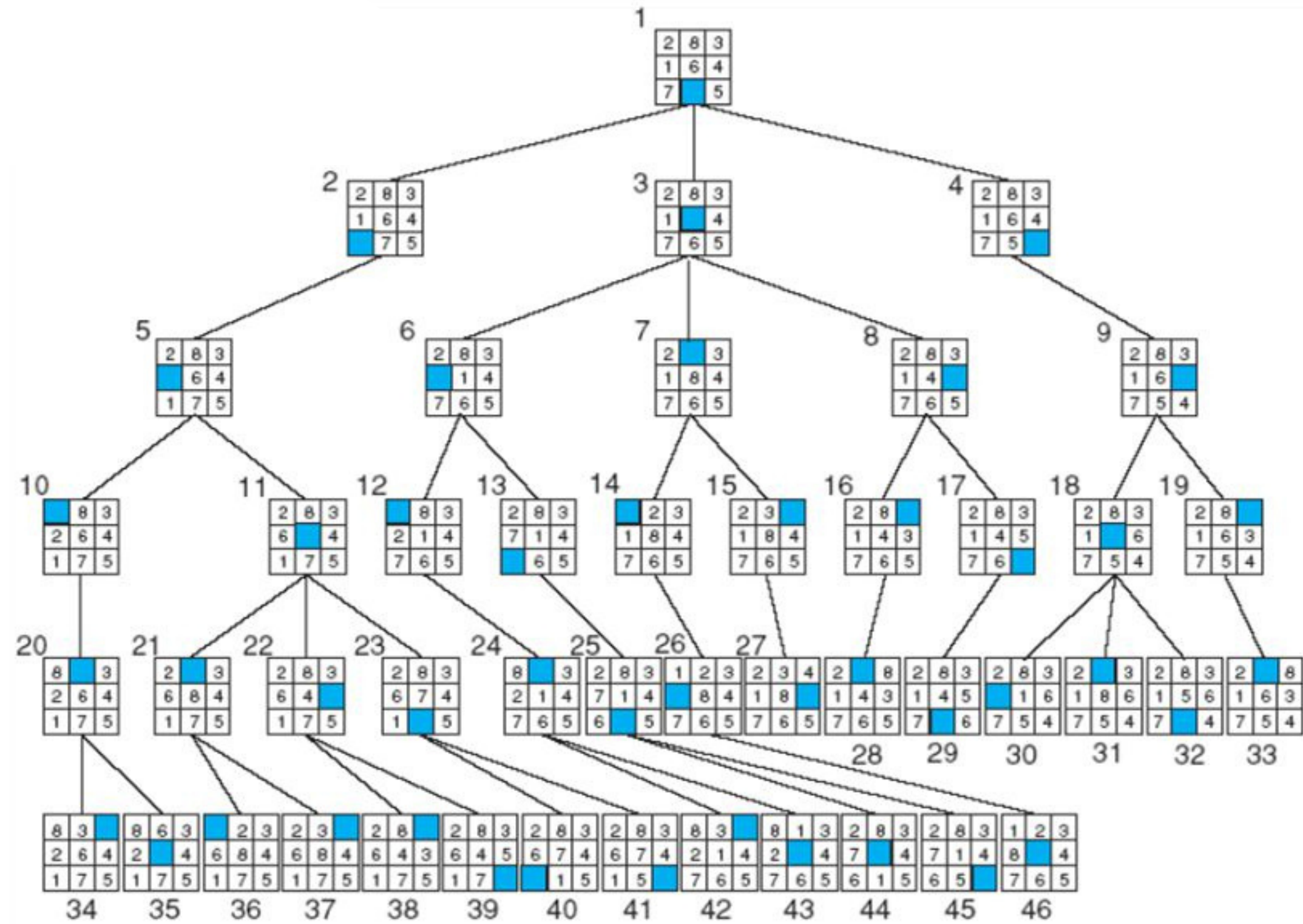
DFS



8 Puzzle

Breadth-First Search

BFS untuk 8-Puzzle




```

def bfs(initial_state, goal_state):
    queue = Queue()
    queue.put(initial_state)
    visited = set()
    path = {tuple(initial_state): None}

    while not queue.empty():
        current_state = queue.get()

        if current_state == goal_state:
            solution_path = []
            while current_state is not None:
                solution_path.append(current_state)
                current_state = path.get(tuple(current_state))
            solution_path.reverse()
            return solution_path

        visited.add(tuple(current_state))

        for move in legal_moves(current_state):
            new_state = make_move(current_state, move)
            if tuple(new_state) not in visited:
                queue.put(new_state)
                path[tuple(new_state)] = current_state

    return None

```

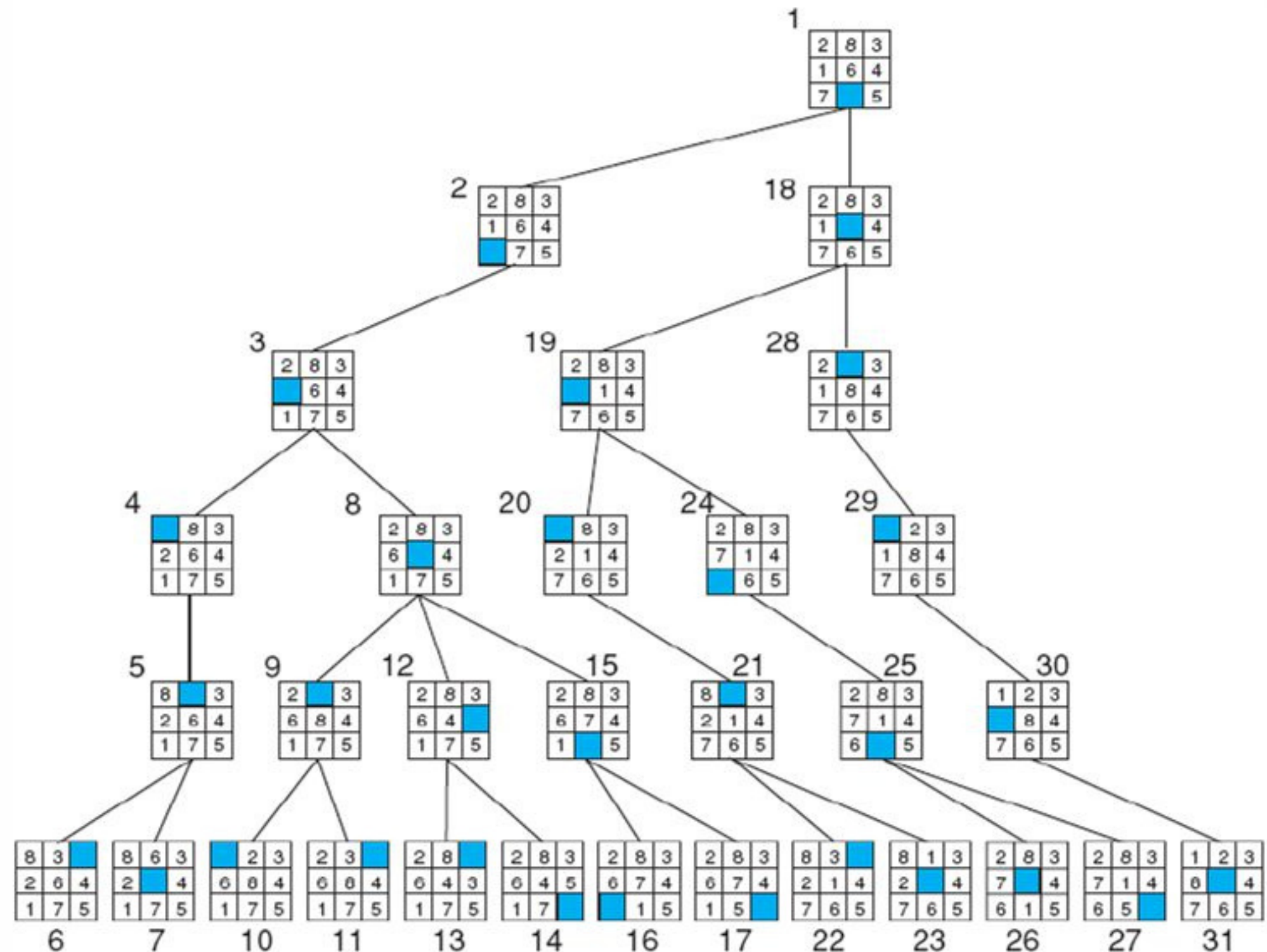
Fungsi BFS pada 8 Puzzle

Antrian queue, himpunan visited, dan kamus path diinisialisasi. Initial_state dimasukkan ke dalam antrian queue dan ditandai sebagai visited. Perulangan dilakukan hingga antrian queue kosong, dengan mengambil node pertama dari antrian, mengecek apakah goal_state sudah ditemukan, dan menambahkan node tetangga yang belum dikunjungi ke antrian.

8 Puzzle

Depth-First Search

DFS untuk 8-Puzzle



```

def dfs(initial_state, goal_state):
    stack = [initial_state]
    visited = set()
    path = {tuple(initial_state): None}

    while stack:
        current_state = stack.pop()

        if current_state == goal_state:
            solution_path = []
            while current_state is not None:
                solution_path.append(current_state)
                current_state = path.get(tuple(current_state))
            solution_path.reverse()
            return solution_path

        visited.add(tuple(current_state))

        for move in legal_moves(current_state):
            new_state = make_move(current_state, move)
            if tuple(new_state) not in visited:
                stack.append(new_state)
                path[tuple(new_state)] = current_state

    return None

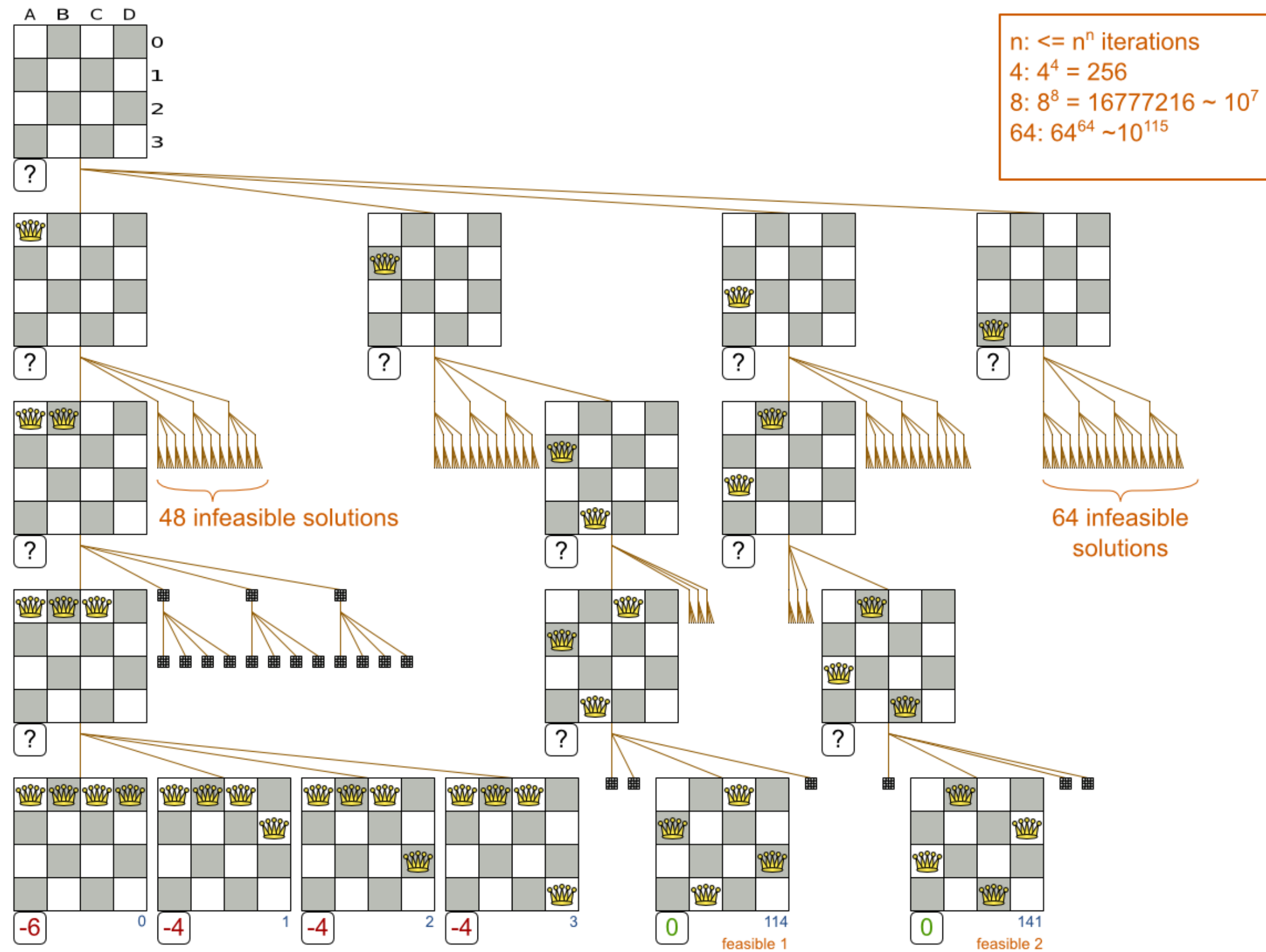
```

Fungsi DFS pada 8 Puzzle

Kode untuk algoritma DFS (depth-first search) yang mengambil state awal dan state tujuan sebagai input, dan mencari solusi dengan mengunjungi setiap node di pohon pencarian secara depth-first, menggunakan stack sebagai struktur data untuk melacak jalur pencarian dan set untuk menghindari mengunjungi node yang sama dua kali.

N Queen

Breadth-First Search



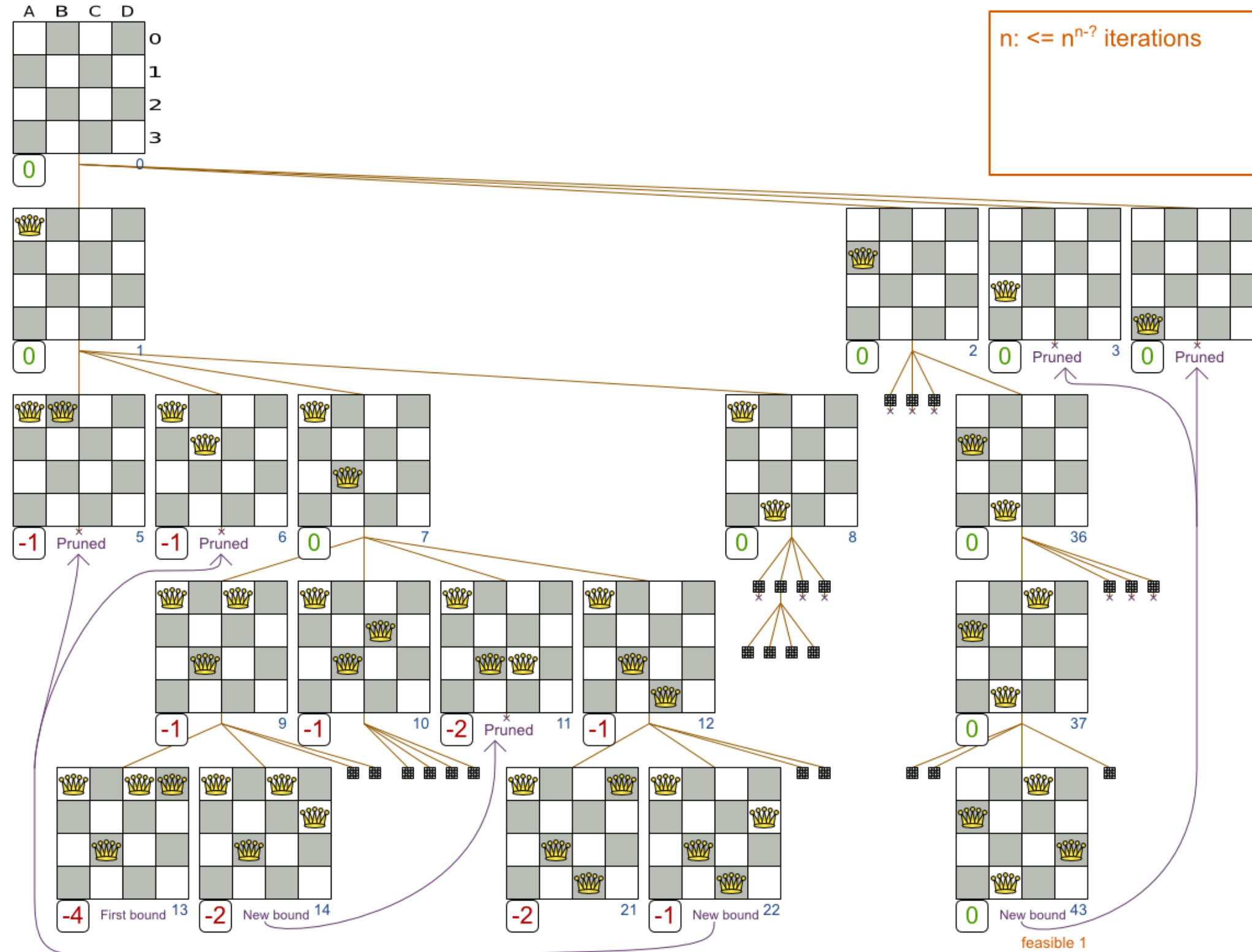
Fungsi BFS pada 8 Queens

Fungsi di atas menyelesaikan masalah n-queens menggunakan algoritma BFS. Fungsi tersebut melakukan iterasi pada sebuah queue yang berisi board kosong dan baris pertama, dan secara iteratif menambahkan kemungkinan solusi pada board hingga ditemukan solusi atau queue kosong. Setiap iterasi, fungsi menempatkan ratu pada setiap kolom pada baris saat ini yang memenuhi aturan, dan menambahkan kemungkinan solusi baru ke queue. Jika tidak ada solusi ditemukan, fungsi mengembalikan None.

```
def solve_n_queens_bfs(n):  
    # inisialisasi queue dengan board kosong dan baris pertama  
    queue = Queue()  
    queue.put(([], 0))  
  
    # iterasi dari queue  
    while not queue.empty():  
        board, row = queue.get()  
  
        # jika semua Queen sudah ditempatkan, return board  
        if row == n:  
            return board  
  
        # menempatkan Queen pada kolom dari baris saat ini  
        for col in range(n):  
            if is_valid(board, row, col):  
                queue.put((board + [col], row + 1))  
  
    # jika tidak ada solusi ditemukan, return none  
    return None
```

N Queen

Depth-First Search



Fungsi DFS pada 8 Queens

Fungsi tersebut memanggil sebuah fungsi rekursif untuk menempatkan ratu pada setiap baris dari board secara iteratif, dan menambahkan ratu pada setiap kolom pada baris saat ini yang memenuhi aturan. Jika solusi ditemukan, fungsi mengembalikan board yang berisi solusi. Jika tidak, fungsi akan menghapus ratu yang telah ditambahkan dari board dan memeriksa kemungkinan solusi lainnya.

```
def solve_n_queens_dfs(n):
    # fungsi rekursif untuk menempatkan Queen pada setiap baris
    def place_queen(board, row):
        # jika semua Queen sudah ditempatkan, return board
        if row == n:
            return board

    # menempatkan Queen pada setiap kolom pada baris saat ini
    for col in range(n):
        if is_valid(board, row, col):
            # menambahkan Queen ke board dan memanggil fungsi rekursif pada baris selanjutnya
            board.append(col)
            result = place_queen(board, row + 1)
            # jika solusi ditemukan, return board
            if result:
                return result
            # jika tidak, hapus Queen yang telah ditambahkan dari board
            board.pop()

    # panggil fungsi rekursif dengan board kosong dan baris pertama
    return place_queen([], 0)
```



TERIMA KASIH :D

Email

shafa@keluargaberencana.com

Website

www.keluargaberencana.com

Phone number

123-456-7890