Trabalho 2 - Análise de Algoritmos

Pedro Lopes - 1911346

Jeronimo Augusto Soares - 1921209

Tarefa 1: Criação do grafo de espaço de estados

In [ ]:
```
from gera_grafo_estados import gera_grafo, imprime_estado
from bfs import BFS_ENV
```

In [ ]:
```
print("Gerando grafo de estados.")
grafo = gera_grafo()
print("Grafo de estados gerado")
```

```
Gerando grafo de estados.
Grafo de estados gerado
```

1. Quantos nós e aresta existem no grafo do espaço

`n` é o número de nós.

`m` é o número de arestas.

In [ ]:
```
n = len(grafo)
m = 0
for no in grafo:
    m+=len(grafo[no])
m //= 2

print(f"{n=}\n{m=}")
```

```
n=362880
m=483840
```

2. Um exemplo de dois nós no grafo conectados por uma aresta

In [ ]:
```
no1 = [1,2,3,4,5,6,7,8,9]
no2 = [1,2,3,4,5,9,7,8,6]

imprime_estado(no1)
imprime_estado(no2)
```

```
---------
| 1 2 3 |
| 4 5 6 |
| 7 8   |
---------
---------
| 1 2 3 |
| 4 5   |
| 7 8 6 |
---------
```

3. Um exemplo de dois nós no grafo que não tem um aresta entre eles

In [ ]:
```
no1 = [1,2,3,4,5,6,7,8,9]
no2 = [9,8,7,6,5,4,3,2,1]

imprime_estado(no1)
imprime_estado(no2)
```

```
---------
| 1 2 3 |
| 4 5 6 |
| 7 8   |
---------
---------
|   8 7 |
| 6 5 4 |
| 3 2 1 |
---------
```

Tarefa 2: Implementação de BFS e contagem de componentes conexos

1. Cóodigo principal da sua BFS

```
def _BFS(self,  start_node):
        start_node = str(start_node)
        i = 1
        l = []
        l.append([start_node])
        self.parents[start_node] = None
        self.visited.add(start_node)
        while True:
            l.append([])
            for node in l[i-1]:
                for neighbor in self.grafo[node]:
                    if neighbor not in self.visited:
                        l[i].append(neighbor)
                        self.parents[neighbor] = node
                        self.visited.add(neighbor)
            if len(l[i]) == 0:
                return l
            i+=1
```

2. Quantidade de componentes conexos

In [ ]:
```
bfs_env = BFS_ENV(grafo)

qtd = bfs_env.get_qtd_componentes_conexos()

print(f"Quantidade de componentes conexos: {qtd}")
```

```
Quantidade de componentes conexos: 2
```

Tarefa 3: Caminhos mais curto

In [ ]:
```
cfg_final = [1,2,3,4,5,6,7,8,9]

imprime_estado(cfg_final)
```

```
---------
| 1 2 3 |
| 4 5 6 |
| 7 8   |
---------
```

1. Configuração inicial viável mais difícil

A configuração inicial viável que necessita o maior núumero de movimentos para se chegar a configuração final:

In [ ]:
```python
cfgs = bfs_env.get_cfg_mais_dificeis(cfg_final)

for cfg in cfgs:
    imprime_estado(cfg)
```

```
---------
| 8 6 7 |
| 2 5 4 |
| 3   1 |
---------
---------
| 6 4 7 |
| 8 5   |
| 3 2 1 |
---------
```

Observamos que há duas configurações iniciais com o mesmo tamanho de caminho mais curto da configuração final tal que este tamanho é máximo.

2. Tamanho do maior menor caminho

Número de movimentos necessários para ir da configuração inicial para a final

In [ ]:
```python
# Escolhemos a primeira configuração dentre as possíveis candidatas
cfg_inicial = cfgs[0]

caminho = bfs_env.get_menor_caminho(cfg_inicial, cfg_final)

# Observe que o primeiro estado também está no caminho,
# logo o tamanho é quantidade de nós do caminho menos 1
tamanho = len(caminho) - 1

print(f"Tamanho do caminho mais difícil: {tamanho}")

for estado in caminho:
    imprime_estado(estado)
```

```
Tamanho do caminho mais difícil: 31
---------
| 8 6 7 |
| 2 5 4 |
| 3   1 |
---------
---------
| 8 6 7 |
| 2   4 |
| 3 5 1 |
---------
---------
| 8   7 |
| 2 6 4 |
| 3 5 1 |
---------
---------
|   8 7 |
| 2 6 4 |
| 3 5 1 |
---------
---------
| 2 8 7 |
|   6 4 |
| 3 5 1 |
---------
---------
| 2 8 7 |
| 3 6 4 |
|   5 1 |
---------
---------
| 2 8 7 |
| 3 6 4 |
| 5   1 |
---------
---------
| 2 8 7 |
| 3 6 4 |
| 5 1   |
---------
---------
| 2 8 7 |
| 3 6   |
| 5 1 4 |
---------
---------
| 2 8   |
| 3 6 7 |
| 5 1 4 |
---------
---------
| 2   8 |
| 3 6 7 |
| 5 1 4 |
---------
---------
| 2 6 8 |
| 3   7 |
| 5 1 4 |
---------
---------
| 2 6 8 |
|   3 7 |
| 5 1 4 |
---------
```

```
---------
| 2 6 8 |
| 5 3 7 |
|   1 4 |
---------
---------
| 2 6 8 |
| 5 3 7 |
| 1   4 |
---------
---------
| 2 6 8 |
| 5 3 7 |
| 1 4   |
---------
---------
| 2 6 8 |
| 5 3   |
| 1 4 7 |
---------
---------
| 2 6   |
| 5 3 8 |
| 1 4 7 |
---------
---------
| 2   6 |
| 5 3 8 |
| 1 4 7 |
---------
---------
| 2 3 6 |
| 5   8 |
| 1 4 7 |
---------
---------
| 2 3 6 |
|   5 8 |
| 1 4 7 |
---------
---------
| 2 3 6 |
| 1 5 8 |
|   4 7 |
---------
---------
| 2 3 6 |
| 1 5 8 |
| 4   7 |
---------
---------
| 2 3 6 |
| 1 5 8 |
| 4 7   |
---------
---------
| 2 3 6 |
| 1 5   |
| 4 7 8 |
---------
---------
| 2 3   |
| 1 5 6 |
| 4 7 8 |
---------
---------
```

```
| 2   3 |
| 1 5 6 |
| 4 7 8 |
---------
---------
|   2 3 |
| 1 5 6 |
| 4 7 8 |
---------
---------
| 1 2 3 |
|   5 6 |
| 4 7 8 |
---------
---------
| 1 2 3 |
| 4 5 6 |
|   7 8 |
---------
---------
| 1 2 3 |
| 4 5 6 |
| 7   8 |
---------
---------
| 1 2 3 |
| 4 5 6 |
| 7 8   |
---------
```