Trabalho sobre Métodos de busca (2020-1)

Aluno: Bruno Aurélio Rôzza de Moura Campos

1. Qual a representação (estrutura de dados) do estado;

- Um estado como este

```
[0, 1, 2]
[3, 4, 5]
[6, 7, 8]
```

é armazenado em uma matriz. Em Python utilizei listas para representar a matriz.

2. Qual a estrutura de dados para a fronteira e nodos fechados;

- Para armazenar a fronteira, decidi colocar em uma lista dentro da matriz de estados. Então ficou assim: `tabuleiro_inicial = [[1, 2, 3], [4, 0, 5], [8, 6, 7], [0, None]]`, sendo a última lista onde armazeno a fronteira[0] e mais uma referência para o node pai.

3. Descrição da implementação (ideia geral e métodos relacionados) das heurísticas

- A implementação foi feita utilizando Python3.
- Para executar:

```
python3 busca.py


# Estado inicial
# Fronteira: 0
# [1, 2, 3]
# [4, 0, 5]
# [8, 6, 7]

# Meta para ganhar
# Fronteira: 0
# [0, 1, 2]
# [3, 4, 5]
# [6, 7, 8]

# Algoritmos de busca:
# 1: Custo Uniforme (sem heurística)
# 2: A* com heurística simples
# 3: A* com heurística de Distância Manhattan
```

- A idea geral é que o usuário escolha um método para solucionar o problema e em seguida o script entra numa iteração para expandir os nodos até encontrar a solução. No final é exibido o caminho

para a melhor solução.
- Métodos
    - gerador_estado_aleatorio() - não utilizado
    - eh_estado_final()
    - eh_repetido_tabuleiro()
    - imprime_tabuleiro()
    - expandir()
    - imprime_caminho()
    - h1() - heurística: número de posições desordenadas
    - h2() - heurística: distância de Manhattan
    - buscar() - principal

4. Como foi gerenciada a fronteira, verificações, quais etapas foram feitas ao adicionar um estado na fronteira (explicação das estratégias, respectivos métodos e possibilidades além do que foi implementado);

Dentro da matriz de estado adicionei uma quarta linha (lista) para armazenar dados como a fronteira e a referência para o nodo pai.

```
tab[3][0] # fronteira
tab[3][1] # referencia para o nodo pai
```

As iterações da fronteira são feitas na função `expandir()`

5. Quais os métodos principais e breve descrição do fluxo do algoritmo;

- A função de `buscar()` funciona como a uma classe driver/run, onde executa toda a lógica. Decidi não inserir na função `main()` para poder gerar uma saída no terminal ao executar o script.
  No início desta função é criado as filas e contadores e depois feito uma iteração sobre o tamanho total da fila. É exigido 2 parâmetros que são: *tipo_busca* e tabuleiro_inicial_. Na principal iteração, é verificado se o estado atual `eh_estado_final()` caso não seja é verificado qual o *tipo_busca*. A partir do *tipo_busca* é entrado em alguma condição, a qual executará a o método de busca (custo_uniforme, heuristica_simples, heuristica_distancia). A partir da definição do *tipo_busca* faço a expansão dos nodos através do método `expandir()`

- A funcão `h1()` é a primeira herística onde analisa o número de peças fora do lugar.

- A função `h2()` é a segunda heurística onde calcula a distância de Manhattan.

6. Caso algum dos objetivos não tenha sido alcançado explique o que você faria VS o que foi feito e exatamente qual o(s) problema(s) encontrado(s), bem como limitações da implementação.

O scripts pode ser mais otimizado, como por exemplo, usando orientação à objetos e tambem estrutura de dados do tipo `set()`.

Escolhi a heurítica da distância de Manhattan(quadras urbanas). Conforme mencionado no livro [2], essa heurística calcula a soma das distâncias dos blocos de suas posições objetivos.

Referências

- [1] vídeos das aulas de INE5633-07238 (20201) - Sistemas Inteligentes
- [2] livro Inteligcia Artificial, Rusell e Norvig
- [3] https://github.com/JaneHJY/8_puzzle
- [4] https://github.com/speix/8-puzzle-solver
- [5] https://gist.github.com/flatline/838202
