Resolução de Problemas por Meio de Busca

Francisco Felipe da Silva, Iury Queiros Sores
Matriculas: 374850, 374852
felipesilva543@alu.ufc.br, iuryqueiros@gmail.com

Introdução:

Este trabalho mostra a implementação de um sistema que resolve o problema proposto do mapa rodoviário da Romênia, com a implementação de três estratégias de busca, busca em largura, busca por custo uniforme, e a busca em profundidade.

Descrição da formulação do problema:

Temos como formulação do problema o mapa da Romênia, e a origem e o destino que se deseja percorrer, neste problema temos Arade como origem e Buchareste como destino.

1. **Estado inicial do problema:**
   Cidade inicial, Arad.
2. **Ações que agente pode executar:**
   Transitar de uma cidade à outra.
   Ir(x, y, z); Ir de x para y no valor de z;
3. **Modelo de transição de estados:**
   O modelo de transição seria basicamente o mapa da Romênia como apresentado no problema, com ligações de cidade e seu respectivos vizinhos.
4. **Teste de objetivo:**
   Verificar se o estado atual é o estado de destino desejado, no caso Bucharest.
5. **Custo de caminho:**

O custo de caminho seria o menor valor em quilômetros da distância entre duas cidades passando pelas demais na busca por custo uniforme, para as outras buscas, o valor é zero pois é irrelevante a distância.

Descrição da implementação dos algoritmos de busca:

A implementação do programa proposto se deu início em ver como iríamos representar o mapa da Romênia, usamos o recurso de tabela de adjacentes, onde representamos cada cidade do mapa no índice da tabela.

Usamos o recurso de enumeração em C++, para numeramos cada cidade.

Além das classes propostas, Node, Action e State, foram criadas uma classe para cada modo de busca, bsf, dfs e ucs, e classes auxiliares como que uma que transforma a entrada de texto para número.

Na criação de cada classe de busca, tivemos a implementação de uma pilha (stack) para a busca em profundidade, uma lista (queue) para a busca em largura, e para a busca em profundidade usamos uma lista com prioridade (priority_queue) que ordenava pelo que

tiver menor custo, usamos o pseudocódigo apresentado na tarefa para a criação da lógica para cada método de busca.

A solução apresentada pelo programa é uma lista de Node, e é apresentado a sequência das cidades que foi visitadas.

Descrição dos experimentos:

A plataforma de software utilizada foi a IDE Qt Creator, com base na linguagem de programação C++, a plataforma de hardware foi um notebook casual com configurações Notebook E5-571G-57MJ Intel Core 5 i5 4GB, 2GB Memória Dedicada 1TB LED 15,6 W8.1 Chumbo - Acer.

Temos como entrada do programa informações como 1 para busca em profundidade, 2 para busca em largura e 3 para busca por custo uniforme, em sequência o estado de origem e destino.

Exp:

```
$ time ./Search 1 ARAD BUCHAREST
Busca em profundidade:
Solução: [ 2 7 10 ]
real    0m0.048s
user    0m0.000s
sys     0m0.000s
```

Abaixo temos imagens de alguns experimentos realizados com as três formas de busca.

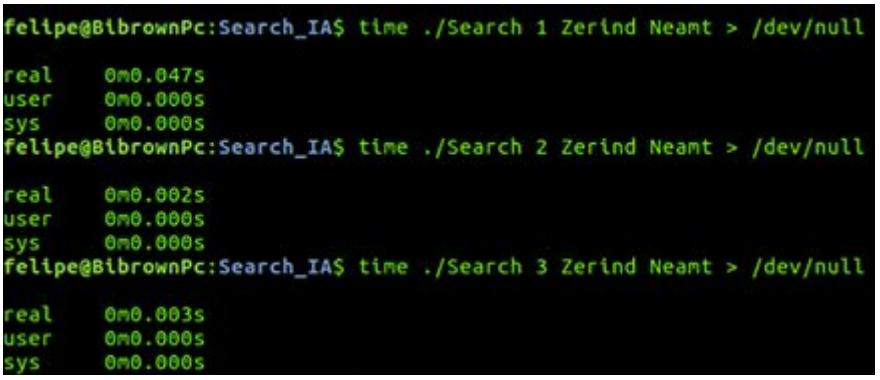
Experimento 1: De Arad para Giurgiu

```
felipe@BibrownPc:Search_IA$ time ./Search 1 Arad Giurgiu > /dev/null

real    0m0.002s
user    0m0.000s
sys     0m0.000s
felipe@BibrownPc:Search_IA$ time ./Search 2 Arad Giurgiu > /dev/null

real    0m0.002s
user    0m0.000s
sys     0m0.000s
felipe@BibrownPc:Search_IA$ time ./Search 3 Arad Giurgiu > /dev/null

real    0m0.002s
user    0m0.000s
sys     0m0.000s
```

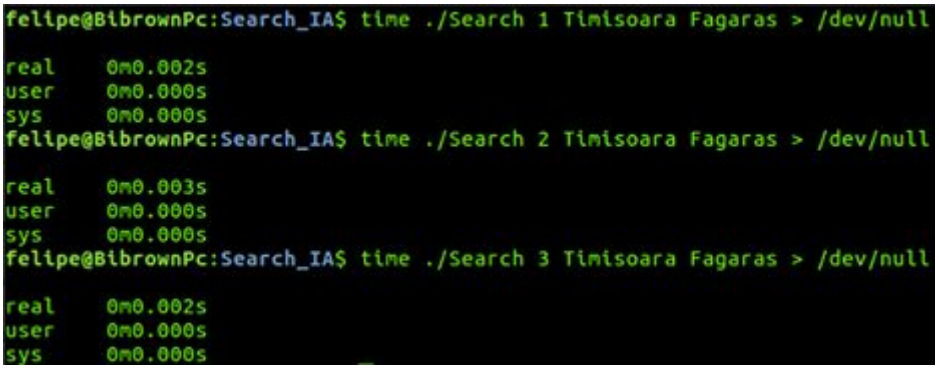
Experimento 2: De Oradea para Eforie

```
felipe@BibrownPc:Search_IA$ time ./Search 1 Oradea Eforie > /dev/null

real    0m0.057s
user    0m0.000s
sys     0m0.000s
felipe@BibrownPc:Search_IA$ time ./Search 2 Oradea Eforie > /dev/null

real    0m0.002s
user    0m0.000s
sys     0m0.000s
felipe@BibrownPc:Search_IA$ time ./Search 3 Oradea Eforie > /dev/null

real    0m0.002s
user    0m0.004s
sys     0m0.000s
```

Experimento 3: De Zerind para Neamt

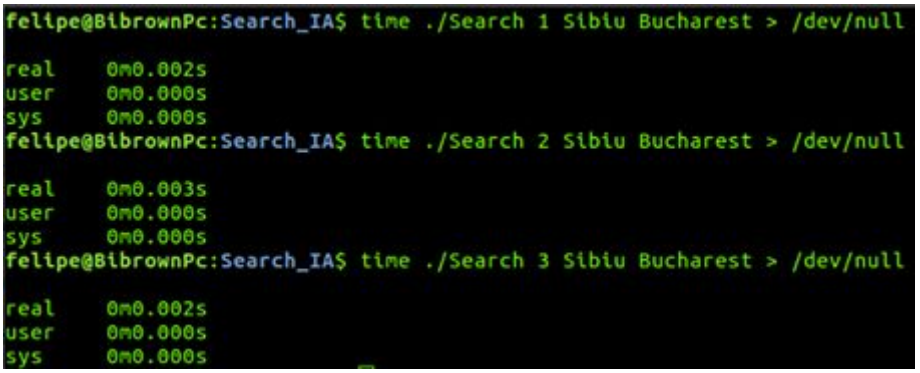


Experimento 4: De Timisoara para Fagaras

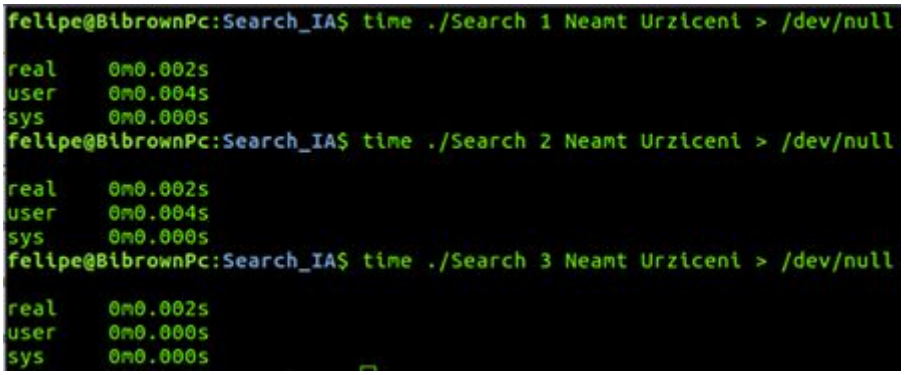


Experimento 5: De Drobeta para Pitesti

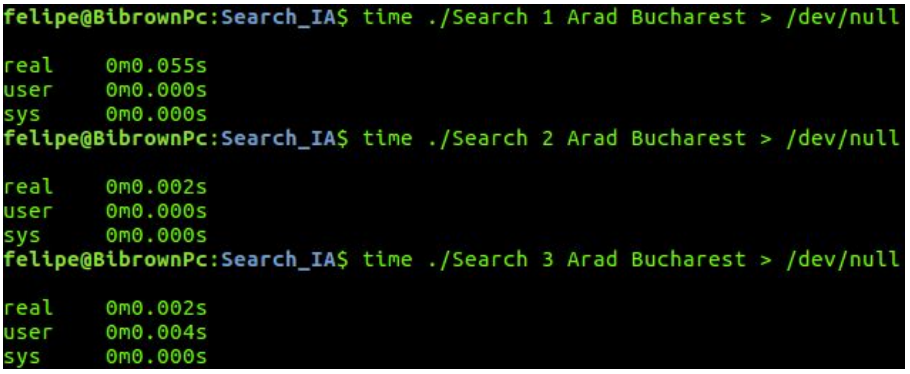
Experimento 6: De Sibiu para Bucharest



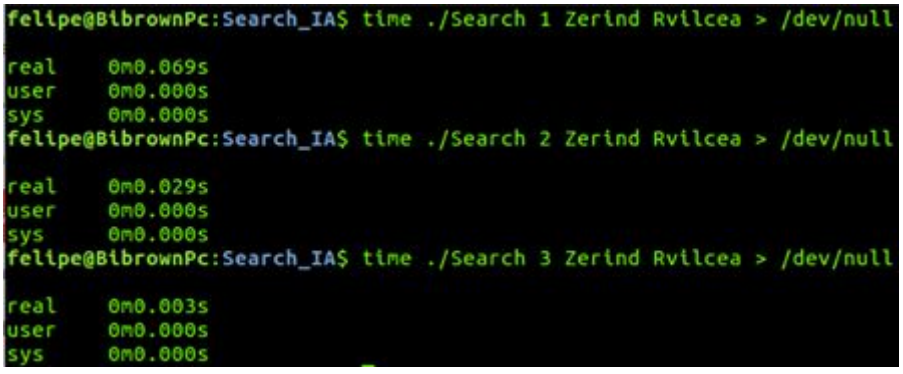
Experimento 7: De Neamt para Urziceni



Experimento 8: De Arad para Bucharest



Experimento 9: De Zerind para Rvilcea



Experimento 10: De Piteste para Urziceni

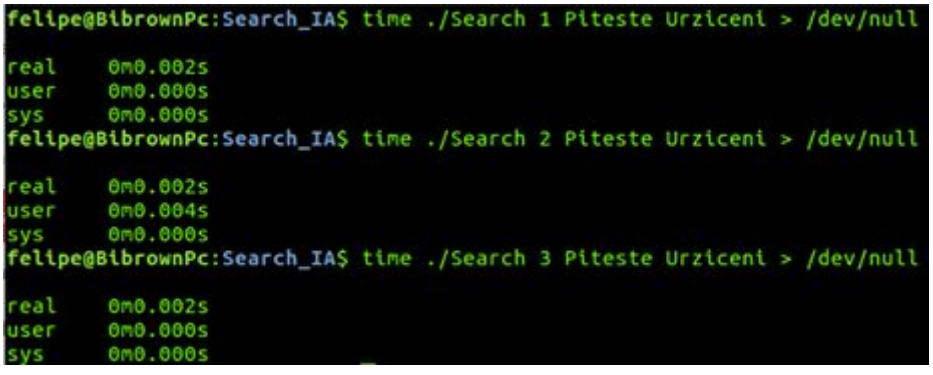
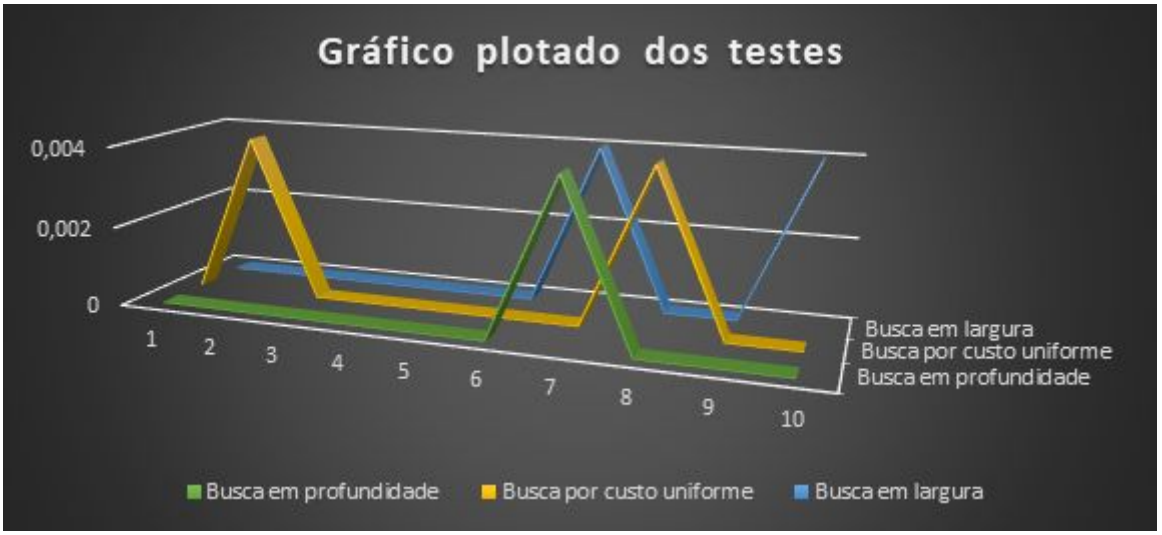


Gráfico de tempo plotado:



O gráfico mostra o tempo(em segundos) de execução dos 3 tipos de busca. Pode-se notar que em alguns experimentos teve uma pequena variação no tempo de execução.

Conclusão:

Não pode-se obter uma resposta de qual algoritmo de busca foi melhor para resolver o problema principal com origem em Arad com destino para Bucharest, experimento 8 nos testes, pois o tempo execução utilizando o comando time não foi suficientemente preciso nas casas decimais e em dois testes foi mostrado tempo de execução igual a zero, logo não foi possível obter uma diferença na execução.