**Alunos:** Wilker Ribeiro de Oliveira dos Santos **CPD:** 26806

Jefferson Vinícius Rodrigues Rabello 31040

Francisca Leomara Morais de Abreu 29800

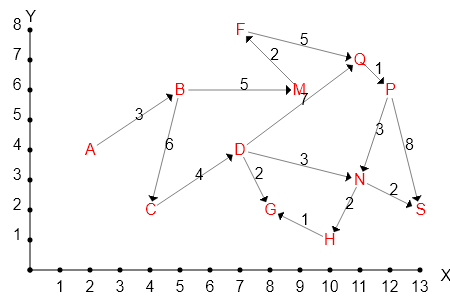
**INTELIGÊNCIA ARTIFICIAL**

**ALGORITMOS PARA BUSCAS**

**Problema:**

O problema escolhido foi uma busca de caminho entre um ponto inicial até um ponto final em um grafo representado num plano cartesiano. Como exemplo, utilizamos o seguinte grafo com os seguintes pontos, sendo que o objetivo é sair do ponto inicial A, para o ponto final S.

Figura 1 – Grafo do problema gerado pela ferramenta desenvolvida



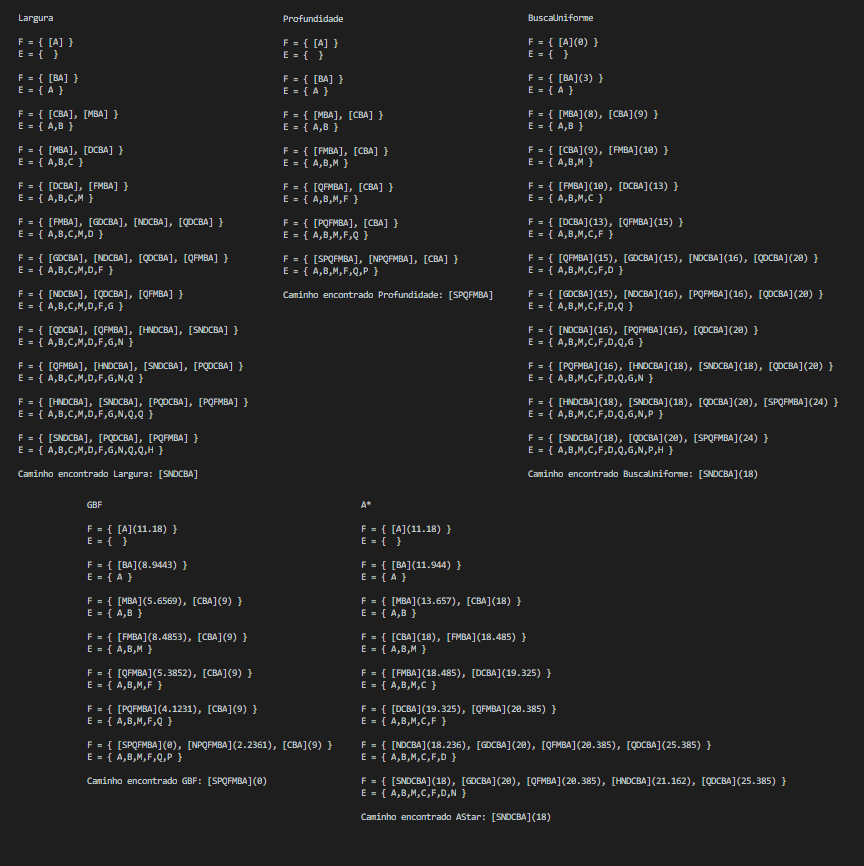
**Solução**

Foi implementada uma API para a definição do grafo e exportação do grafo em SVG. Foram implementados algoritmos para: Busca em Largura, Busca em Profundidade, Busca de Custo Uniforme, Busca GBF e Busca A\*, com base no grafo definido.

Executando os algoritmos implementados, foi escolhida a Busca A\* pois foi a que retornou o caminho com menor custo em menos passos, permitindo tanto custo de caminho entre pontos quanto valores de heurísticas, sendo assim, uma busca mais completa.

A figura a seguir representa o resultado da execução de todas as buscas implementadas, utilizando o mesmo conjunto de dados apresentado no problema.

Figura 2 - Resultado das buscas



O código desenvolvido está disponível nos seguintes links:

<https://c9.io/wilker7ribeiro/algoritmos-busca>

<https://github.com/wilker7ribeiro/algoritmos-buscas>

**Busca A\***

A busca A\* é bastante utilizada por sua performance e precisão. É uma solução que tenta maximizar a performance combinando elementos da busca de custo uniforme e da busca GBF (gulosa).

Sua função de avaliação leva em consideração o valor do caminho do nó inicial até o nó atual, e valor de heurísticas como distância entre o nó atual e o nó final.

Desta forma, o algoritmo fica mais inteligente, executando menos passos para encontrar a solução.

O passo a passo a seguir representa o script da busca A\*:

1. Primeiramente são realizados os cálculos das heurísticas para cada ponto e os valores são salvos para cada ponto para futuramente serem utilizados.
2. É inicializado um grupo de valores a serem explorados (fronteira) e de valores já explorados.
3. Coloca-se o ponto inicial na fronteira.
4. Enquanto a fronteira não estiver vazia, repita:
   1. Explora o primeiro item da fronteira.
      1. Se o item for o item final (objetivo) retorna o caminho e finaliza o algoritmo.
      2. Se não:
         1. O item explorado é removido da fronteira e adicionado nos explorados.
         2. Para cada item vizinho não explorado do item explorado, é calculado o custo total (custo de caminho + heurística), e são inseridos na fronteira, de modo que a fronteira esteja sempre ordenada conforme o custo total em modo ascendente.
5. Caminho não é encontrado

**Conclusão**

Foi implementado o algoritmo, foi criado uma massa de dados para realizar testes. Foram executados o algoritmo implementado e um teste de mesa manual com a mesma massa de dados. Ambos retornaram os mesmos resultados, provando que o algoritmo implementado funciona:

* **Teste de mesa A\*:**

|  |  |
| --- | --- |
| **PONTO** | **h(n)** |
| **S** | 0 |
| **A** | 11,18 |
| **B** | 8,944 |
| **C** | 9 |
| **D** | 6,325 |
| **F** | 8,485 |
| **G** | 5 |
| **H** | 3,162 |
| **M** | 5,657 |
| **N** | 2,236 |
| **P** | 4,123 |
| **Q** | 5,385 |

F = { **[a](11,18)** }

E = { }

F = { **[ba](12,944)** }

E = { **a** }

F = { **[mba](13,657), [cba](18)** }

E = { a, **b** }

F = { [cba](18) **, [fmba](18,485)**}

E = { a, b, **m** }

F = { [fmba](18,485), **[dcba](19,325)** }

E = { a , b, m, **c** }

F = { [dcba](19,325), **[qfmba](20,385)** }

E = { a , b, m, c, **f** }

F = { **[ndcba](18,236)**, **[gdcba](20)**, [qfmba](20,385), **[qdcba](25,385)** }

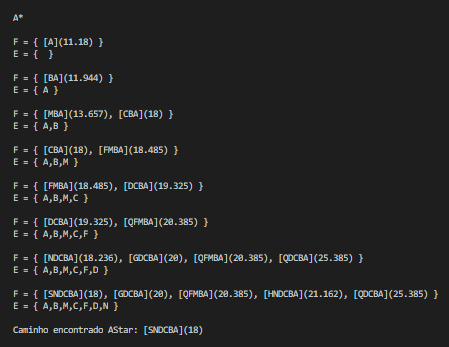
E = { a , b, m, c, f, **d** }

F = { **[sndcba](18), [gdcba](20) ,** [qfmba](20,385), **[hndcba](21,162)** , **[qdcba](25,385)** }

E = { a , b, m, c, f, d, **n** }

**Encontra-se** **[sndcba] onde o custo é 18**

* **Saída do algoritmo:**

****