Capítulo 5: Buscas

Vitor Henrique Carvalho de Morais - 12221ECP011

11 de fevereiro de 2026

Disciplina: Programação Lógica e Inteligência Artificial
Professor(a): Marcelo Rodrigues
Curso: Engenharia de Computação

Introdução

Neste capítulo, estudamos como funcionam as buscas na programação lógica, especialmente utilizando a linguagem Prolog. A ideia principal é entender como um computador pode encontrar soluções para um problema seguindo regras bem definidas, quase como se estivesse explorando diferentes caminhos até chegar à resposta correta. Em vez de dizer exatamente como o programa deve fazer cada passo, na programação lógica nós descrevemos o problema e as regras, e o próprio sistema se encarrega de procurar a solução.

Também comparamos o Prolog com outras linguagens de programação. Em linguagens mais comuns, como JavaScript, o programador precisa escrever passo a passo como a busca será feita. Já no Prolog, o mecanismo de busca faz parte da própria linguagem, funcionando automaticamente por meio de regras e tentativas sucessivas até encontrar uma solução. Essa comparação ajuda a entender as diferenças entre os estilos de programação e mostra como cada abordagem resolve problemas de maneira diferente.

Resumo: Capítulo 5 - Buscas

5.1 - Árvores de Busca

Uma árvore de busca é uma forma de organizar todas as possibilidades para resolver um problema. Imagine que você está tentando encontrar a saída de um labirinto. Você começa em um ponto inicial e, a cada passo, pode escolher diferentes caminhos. Cada escolha leva a novas possibilidades, e essas possibilidades continuam se dividindo até que você encontre a saída ou perceba que aquele caminho não funciona.

A árvore de busca funciona como um desenho dessas escolhas. O ponto onde tudo começa é chamado de raiz. A partir dele, surgem vários caminhos, como galhos de uma árvore. Cada novo lugar que você pode chegar representa uma nova situação do problema. Quando um caminho não pode mais continuar, ele termina, como uma folha da árvore.

Assim, a árvore de busca ajuda o computador a explorar diferentes caminhos de forma organizada até encontrar a solução correta.

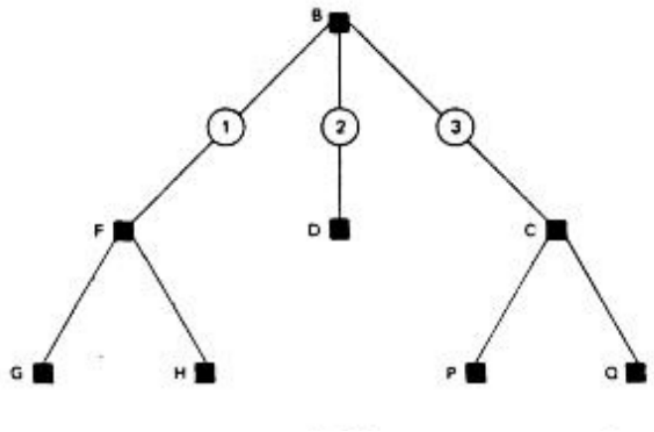

Figura 1: Exemplo de uma árvore de busca.

Imagine que essa imagem é como um mapa de escolhas.

Lá em cima está a letra B, que é o ponto onde tudo começa. É como se você estivesse em um jogo e tivesse que decidir para onde ir primeiro. A partir de B, você tem três caminhos possíveis, representados pelos números 1, 2 e 3. Cada número é uma escolha diferente.

Se você escolhe o caminho 1, você chega na letra F. Depois disso, ainda pode escolher entre dois novos caminhos, que levam até G ou H. Quando você chega em G ou H, o caminho termina, como se fosse o fim daquela tentativa.

Se você escolhe o caminho 2 logo no começo, você chega em D, e ali o caminho também termina.

Se você escolhe o caminho 3, você vai para C, e de lá pode seguir para P ou Q, que também são finais.

Então essa imagem mostra como, a partir de um começo, várias escolhas vão surgindo, como galhos de uma árvore. Cada galho leva a um lugar diferente. A ideia da árvore de busca é justamente mostrar todos os caminhos possíveis que você pode seguir até chegar em um resultado.

5.2 - Busca em Largura

Imagine que você está procurando um tesouro escondido em um castelo cheio de portas. Em vez de sair correndo por um corredor até o fim, a busca em largura funciona assim: você olha primeiro todas as portas que estão bem perto de você. Depois que verifica todas elas, você passa para as portas que ficam um pouco mais longe. Só depois você vai para as ainda mais distantes. Ou seja, você vai explorando "andar por andar", sem se aprofundar demais em um único caminho. Assim, se o tesouro estiver perto do começo, você encontra mais rápido.

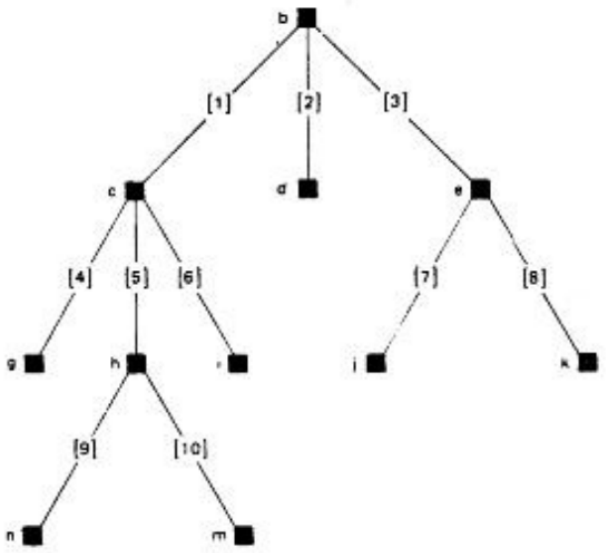

Figura 2: Exemplo de busca em largura.

Para organizar isso, usamos algo chamado fila. Pense em uma fila como a de uma padaria: quem entra primeiro é atendido primeiro. Na busca em largura, cada caminho que ainda precisa ser explorado entra no final da fila. O caminho que entrou primeiro será o primeiro a ser testado.

```
Fila = [ [r(1,c),r(raiz,b)],
         [r(2,d),r(raiz,b)],
         [r(3,e),r(raiz,b)] ].
```

Figura 3: Exemplo de fila para busca em largura.

As filas estendidas são simplesmente a fila principal já com os novos caminhos adicionados no final. Quando você pega um caminho da frente da fila e descobre novos caminhos possíveis a partir dele, você coloca esses novos caminhos no fim da fila. Assim, a fila vai ficando maior, cheia de possibilidades organizadas.

```
FilaEstendida = [[r(2,d),r(raiz,b)],
                 [r(3,e),r(raiz,b)],
                 [r(4,g),r(1,c),r(raiz,b)],
                 [r(5,h),r(1,c),r(raiz,b)],
                 [r(6,i),r(1,c),r(raiz,b)]].
```

Figura 4: Exemplo de fila estendida para busca em largura.

Já a fila de extensões é o conjunto desses novos caminhos que acabaram de nascer

a partir de um caminho que você acabou de explorar. Primeiro você cria essas extensões (os novos caminhos possíveis) e depois coloca todas elas no final da fila principal. Dessa forma, a busca continua organizada, sempre explorando primeiro os caminhos mais antigos antes de passar para os mais recentes.

```
ListaDeExtensões = [[r(4,g), r(1,c), r(raiz,b)],
                    [r(5,h),r(1,c),r(raiz,b)],
                    [r(6,i),r(1,c), r(raiz,b)]].
```

Figura 5: Exemplo de fila de extensões para busca em largura.

Conclusão

Síntese das ideias principais.

Referências

Autor. *Título da obra*. Editora, ano.