Documentação do Algoritmo de Torre de Hanoi

Marcos Vinicius Silva

28 de Setembro de 2023

1 Introdução

O algoritmo da Torre de Hanoi é um problema clássico de divisão e conquista que envolve a transferência de discos entre três hastes. Este documento detalha o código do algoritmo e destaca o uso da indução na solução do problema.

2 Algoritmo de Torre de Hanoi

O algoritmo de Torre de Hanoi é implementado na função `torreDeHanoi`. Esta função utiliza a técnica de recursão e indução para mover os discos. Ela toma quatro argumentos:

- `n`: O número de discos a serem movidos.
- `hasteOrigem`: A haste de origem da qual os discos devem ser movidos.
- `hasteAuxiliar`: A haste auxiliar usada como intermediária na transferência.
- `hasteDestino`: A haste de destino para onde os discos devem ser movidos.

2.1 Recursão e Indução

A indução desempenha um papel fundamental no algoritmo da Torre de Hanoi. Vamos detalhar como a indução é usada neste algoritmo:

```
function torreDeHanoi(n, hasteOrigem, hasteAuxiliar,
    hasteDestino) {
  if (n === 1) {
    let disco = torres[hasteOrigem].pop();
    torres[hasteDestino].push(disco);
    console.log(`Mova o disco ${disco} de ${hasteOrigem}
        para ${hasteDestino}`);
    imprimirEstadoTorres();
    return;
  }
```

```
  torreDeHanoi(n - 1, hasteOrigem, hasteDestino,
      hasteAuxiliar);
  let disco = torres[hasteOrigem].pop();
  torres[hasteDestino].push(disco);
  console.log(`Mova o disco ${disco} de ${hasteOrigem} para
      ${hasteDestino}`);
  imprimirEstadoTorres();
  torreDeHanoi(n - 1, hasteAuxiliar, hasteOrigem,
      hasteDestino);
}
```

Aqui, a indução começa com a verificação do caso base: quando `n === 1`. Nesse caso, o algoritmo move o disco diretamente da haste de origem para a haste de destino e imprime uma mensagem. Isso representa a base da indução, onde o problema é resolvido para o caso mais simples.

Quando `n` não é igual a 1, o algoritmo entra na parte recursiva, que é a aplicação da indução. O algoritmo divide o problema em três etapas:

a. **Mover ‘n-1‘ discos da haste de origem para a haste auxiliar**: - Neste passo, uma chamada recursiva é feita para mover `n-1` discos da haste de origem para a haste auxiliar, usando a haste de destino como auxiliar.

b. **Mover o disco restante da haste de origem para a haste de destino**: - Após mover os `n-1` discos para a haste auxiliar, o algoritmo move o disco restante (o maior disco) da haste de origem para a haste de destino.

c. **Mover os ‘n-1‘ discos da haste auxiliar para a haste de destino**: - Finalmente, outra chamada recursiva é feita para mover os `n-1` discos da haste auxiliar para a haste de destino, usando a haste de origem como auxiliar.

Isso demonstra o princípio da indução, onde o algoritmo funciona para um caso menor (`n-1` discos) e, com base nessa suposição, prova que também funcionará para um caso maior (`n` discos). Esse processo se repete até que todos os discos tenham sido movidos para a haste de destino.

3 Função de Impressão do Estado das Torres

```
function imprimirEstadoTorres() {
  console.log('--- Resultado das Torres ---');
  console.log('Haste A: ' + torres['A'].join(' '));
  console.log('Haste B: ' + torres['B'].join(' '));
  console.log('Haste C: ' + torres['C'].join(' '));
  console.log('------------------------------');
  console.log('');
}
```