BUSCA SEQUENCIAL E BUSCA BINÁRIA

Introdução:

A busca é uma operação fundamental em ciência da computação e é usada em muitos algoritmos. A busca sequencial e a busca binária são dois algoritmos comuns de busca que você encontrará frequentemente na programação. Nesta apostila, veremos como implementar esses algoritmos em JavaScript.

Busca Sequencial:

A busca sequencial é um algoritmo simples que percorre toda a lista em busca do elemento desejado. Ele começa na primeira posição e verifica cada elemento até encontrar o elemento que está procurando. Se o elemento não estiver na lista, a busca sequencial percorrerá toda a lista. O algoritmo de busca sequencial tem uma complexidade de tempo O(n), onde n é o tamanho da lista.

Exemplo de código:

```
1 function buscaSequencial(lista, item) {
2   for (let i = 0; i < lista.length; i++) {
3     if (lista[i] === item) {
4       return i;
5     }
6   }
7   return -1;
8 }
9
10 let lista = [2, 5, 8, 12, 16];
11 console.log(buscaSequencial(lista, 8)); // retorna 2
12 console.log(buscaSequencial(lista, 10)); // retorna -1
```

Neste exemplo, a função buscaSequencial recebe uma lista e um item para procurar na lista. A função percorre cada elemento da lista usando um loop for. Se o elemento da lista for igual ao item procurado, a função retorna o índice desse elemento na lista. Se o elemento não for encontrado na lista, a função retorna -1.

Busca Binária:

A busca binária é um algoritmo de busca mais eficiente que a busca sequencial, especialmente para listas grandes. A busca binária requer que a lista esteja ordenada antes de começar a busca. Ele começa no meio da lista e verifica se o elemento desejado é maior ou menor do que o elemento atual. Se o elemento desejado for menor, a busca continua na metade inferior da lista. Se o elemento desejado for maior, a busca continua na metade superior da lista. Esse processo é repetido até que o elemento seja encontrado ou até que a busca alcance o final da lista. O algoritmo de busca binária tem uma complexidade de tempo O(log n), onde n é o tamanho da lista.

Exemplo de código:

```
function buscaBinaria(lista, item) {
  let inicio = 0;
  let fim = lista.length - 1;
  let meio;

  while (inicio <= fim) {
    meio = Math.floor((inicio + fim) / 2);
    if (lista[meio] === item) {
      return meio;
    } else if (lista[meio] < item) {
      inicio = meio + 1;
    } else {
      fim = meio - 1;
    }
  }
  return -1;
}

let lista = [2, 5, 8, 12, 16];
console.log(buscaBinaria(lista, 8)); // retorna 2
console.log(buscaBinaria(lista, 10)); // retorna -1
```

Neste exemplo, a função buscaBinaria recebe uma lista ordenada e um item para procurar na lista. A função usa um loop while para dividir a lista ao meio e verificar se o elemento desejado é menor ou maior do que o elemento do meio.
Se o elemento desejado for igual ao elemento do meio, a função retorna o índice desse elemento na lista. Se o elemento desejado for menor do que o elemento do meio, a função continua a busca na metade inferior da lista. Se o elemento desejado for maior do que o elemento do meio, a função continua a busca na metade superior da lista. O processo é repetido até que o elemento seja encontrado ou até que a busca alcance o final da lista. Se o elemento não for encontrado na lista, a função retorna -1.

Conclusão:

A busca sequencial e a busca binária são algoritmos comuns de busca que você encontrará frequentemente na programação. A busca sequencial é um algoritmo simples que percorre toda a lista em busca do elemento desejado, enquanto a busca binária é um algoritmo mais eficiente que requer que a lista esteja ordenada antes de começar a busca. Em geral, se você tem uma lista grande e ordenada, é melhor usar a busca binária. Se a lista não for grande ou não estiver ordenada, é melhor usar a busca sequencial.

Espero que esta apostila tenha sido útil para você entender como implementar a busca sequencial e a busca binária em JavaScript. Lembre-se sempre de testar seus algoritmos e verificar se eles estão funcionando corretamente antes de usá-los em um projeto real.