APOSTILA 02 - ORDENAÇÃO DE VETORES (PARTE 2/2)

Introdução

A ordenação de vetores é uma operação muito comum em programação, e há vários algoritmos que podem ser usados para realizar essa tarefa. Nesta apostila, vamos nos concentrar em dois algoritmos populares: o Merge Sort e o Quick Sort.

Ambos os algoritmos são baseados na ideia de dividir e conquistar, ou seja, eles dividem o vetor em pedaços menores, resolvem cada pedaço individualmente e, em seguida, combinam os resultados para obter o vetor ordenado final.

Merge Sort

O Merge Sort é um algoritmo recursivo que divide o vetor ao meio repetidamente até que cada subvetor tenha apenas um elemento. Em seguida, os subvetores são combinados em pares, mesclando os elementos em ordem crescente, até que todo o vetor esteja ordenado.

O algoritmo tem uma complexidade de tempo de O(n log n), o que o torna bastante eficiente para vetores grandes.

Aqui está o código JavaScript para o Merge Sort:

```javascript
function mergeSort(arr) {
  if (arr.length <= 1) {
    return arr;
  }

  const mid = Math.floor(arr.length / 2);
  const left = mergeSort(arr.slice(0, mid));
  const right = mergeSort(arr.slice(mid));

  return merge(left, right);
}

function merge(left, right) {
  const result = [];

  let i = 0;
  let j = 0;

  while (i < left.length && j < right.length) {
    if (left[i] < right[j]) {
      result.push(left[i]);
      i++;
    } else {
      result.push(right[j]);
      j++;
    }
  }

  return result.concat(left.slice(i)).concat(right.slice(j));
}
```

O algoritmo Merge Sort é implementado na função mergeSort, que recebe um vetor como parâmetro e retorna o vetor ordenado. A função mergeSort é recursiva e chama a si mesma para dividir o vetor em subvetores menores.

A função merge é responsável por combinar dois subvetores ordenados em um único vetor ordenado. Ela usa dois ponteiros, i e j, para percorrer os subvetores e adiciona o menor elemento de cada subvetor ao resultado.

Quick Sort

O Quick Sort é outro algoritmo de ordenação baseado na divisão e conquista. Ele escolhe um elemento como pivô e divide o vetor em dois subvetores: um com elementos menores que o pivô e outro com elementos maiores. Em seguida, o algoritmo é aplicado recursivamente a cada subvetor.

O pivô pode ser escolhido de várias maneiras, mas geralmente é o primeiro elemento do vetor. O Quick Sort tem uma complexidade de tempo média de O(n log n), mas pode chegar a $O(n^2)$ no pior caso.

Aqui está o código JavaScript para o Quick Sort:

```javascript
function quickSort(arr, left = 0, right = arr.length - 1) {
  if (left < right) {
    const pivotIndex = partition(arr, left, right);
    quickSort(arr, left, pivotIndex - 1);
    quickSort(arr, pivotIndex + 1, right);
  }
  return arr;
}

function partition(arr, left, right) {
  const pivotIndex = Math.floor((left + right) / 2);
  const pivotValue = arr[pivotIndex];

  let i = left;
  let j = right;

  while (i <= j) {
    while (arr[i] < pivotValue) {
      i++;
    }
    while (arr[j] > pivotValue) {
      j--;
    }
    if (i <= j) {
      swap(arr, i, j);
      i++;
      j--;
    }
  }

  return i;
}

function swap(arr, i, j) {
  const temp = arr[i];
  arr[i] = arr[j];
  arr[j] = temp;
}
```

O algoritmo Quick Sort é implementado na função quickSort, que recebe um vetor como parâmetro e retorna o vetor ordenado. A função quickSort é recursiva e chama a si mesma para ordenar os subvetores.

A função partition é responsável por dividir o vetor em dois subvetores. Ela escolhe o pivô como o elemento do meio do vetor e percorre o vetor, movendo os elementos menores para a esquerda do pivô e os elementos maiores para a direita.

A função swap é usada para trocar elementos de posição no vetor.

Comparação

Tanto o Merge Sort quanto o Quick Sort são algoritmos eficientes para ordenar vetores grandes. No entanto, o Merge Sort é mais estável e previsível em termos de desempenho, enquanto o Quick Sort pode ser mais rápido em alguns casos, mas é menos estável.

O Merge Sort divide o vetor em subvetores iguais, o que garante que o algoritmo sempre terá um desempenho O(n log n) consistente, independentemente dos dados de entrada. Além disso, o Merge Sort é um algoritmo estável, o que significa que ele preserva a ordem relativa dos elementos iguais.

Por outro lado, o Quick Sort tem um desempenho médio O(n log n), mas pode chegar a $O(n^2)$ no pior caso, quando o pivô escolhido é o menor ou o maior elemento do vetor. Além disso, o Quick Sort não é estável, o que significa que ele pode alterar a ordem relativa dos elementos iguais.

Conclusão

Nesta apostila, vimos como implementar os algoritmos Merge Sort e Quick Sort em JavaScript. Ambos os algoritmos são baseados na ideia de dividir e conquistar e podem ser usados para ordenar vetores grandes com eficiência.

O Merge Sort é um algoritmo estável e previsível em termos de desempenho, enquanto o Quick Sort pode ser mais rápido em alguns casos, mas é menos estável. É importante escolher o algoritmo de ordenação certo para cada caso de uso, dependendo das necessidades de desempenho e estabilidade.