Algoritmo de Encontrar Maior Soma de Subarray contido

Marcos Vinicius Silvaa

Dezembro, 2023

1 Introdução

Este relatório apresenta uma explicação técnica do algoritmo para encontrar a maior soma de um subarray contido em um array unidimensional utilizando a abordagem de divisão e conquista em JavaScript.

2 Algoritmo e Implementação

2.1 Objetivo

O objetivo deste algoritmo é encontrar a maior soma de um subarray contido em um array unidimensional.

2.2 Abordagem Utilizada

O algoritmo utiliza a técnica de divisão e conquista para resolver o problema. Ele divide recursivamente o array em subarrays menores, encontra a maior soma de subarrays em cada subdivisão e, por fim, combina as soluções para obter a maior soma do subarray contido.

2.3 Detalhamento do Código

Aqui está o código do algoritmo:

```
function findMaxSubarray(arr) {
  // Caso base: se o array tem apenas um elemento, retorne esse elemento.
  if (arr.length === 1) {
    return arr[0];
  }

  const mid = Math.floor(arr.length / 2);
  const leftHalf = arr.slice(0, mid);
  const rightHalf = arr.slice(mid);
```

```
  const maxLeft = findMaxSubarray(leftHalf);
  const maxRight = findMaxSubarray(rightHalf);

  const maxCrossing = findMaxCrossingSubarray(arr, mid);

  return Math.max(maxLeft, maxRight, maxCrossing);
}

function findMaxCrossingSubarray(arr, mid) {
  let leftSum = -Infinity;
  let sum = 0;

  for (let i = mid; i >= 0; i--) {
    sum += arr[i];
    leftSum = Math.max(leftSum, sum);
  }

  let rightSum = -Infinity;
  sum = 0;

  for (let i = mid + 1; i < arr.length; i++) {
    sum += arr[i];
    rightSum = Math.max(rightSum, sum);
  }

  return leftSum + rightSum;
}

// Exemplo de uso:
const args = process.argv.slice(2);
const array = args.map(arg => parseInt(arg));
console.log("Array fornecido:", array);

const maxSubarraySum = findMaxSubarray(array);
console.log("Maior soma do subarray contido:", maxSubarraySum);
```

2.4 Explicação do Código

2.4.1 `findMaxSubarray`

Esta função é o núcleo do algoritmo. Ela divide o array em duas metades, chama a si mesma recursivamente para cada metade e, então, encontra a maior soma entre a parte esquerda, a parte direita e o subarray que cruza o meio.

2.4.2 `findMaxCrossingSubarray`

Esta função encontra a maior soma de um subarray que cruza o ponto médio do array. Ela itera sobre as partes esquerda e direita do array a partir do ponto médio e determina a maior soma de subarrays nessas regiões.

```
function findMaxCrossingSubarray(arr, mid) {
  let leftSum = -Infinity;
  let sum = 0;

  for (let i = mid; i >= 0; i--) {
    sum += arr[i];
    leftSum = Math.max(leftSum, sum);
  }

  let rightSum = -Infinity;
  sum = 0;

  for (let i = mid + 1; i < arr.length; i++) {
    sum += arr[i];
    rightSum = Math.max(rightSum, sum);
  }

  return leftSum + rightSum;
}
```

2.5 Complexidade

A complexidade do algoritmo é $O(n \log n)$, onde n é o tamanho do array. Isso se deve ao fato de que o algoritmo divide o array pela metade recursivamente.

3 Exemplo de Uso

Um exemplo de uso do algoritmo é fornecido ao executar o script JavaScript a partir da linha de comando, passando os números como argumentos. O resultado será a maior soma do subarray contido.

4 Conclusão

O algoritmo de encontrar a maior soma de um subarray contido utilizando a técnica de divisão e conquista é uma abordagem eficiente para resolver esse problema em arrays unidimensionais.