Phân Tích Bài Toán 3364: Tổng Dương Nhỏ Nhất của Mảng Con

Quang Cảnh

November 30, 2024

1 Đề Bài

Cho một mảng số nguyên `nums` và hai số nguyên `l` và `r`:

- `l`: kích thước nhỏ nhất của mảng con.
- `r`: kích thước lớn nhất của mảng con.

Nhiệm vụ là tìm tổng nhỏ nhất của bất kỳ mảng con nào có tổng dương và kích thước nằm trong khoảng $[l, r]$. Nếu không tồn tại mảng con thỏa mãn, trả về -1.

Ràng Buộc Đầu Vào

- $-10^4 \leq$ `nums[i]` $\leq 10^4$
- $1 \leq l \leq r \leq$ `len(nums)`

Kết Quả Đầu Ra

Trả về tổng nhỏ nhất dương của bất kỳ mảng con nào nằm trong khoảng $[l, r]$, hoặc -1 nếu không có mảng con thỏa mãn.

2 Cách Tiếp Cận

2.1 Tổng Quan

Chúng ta cần xác định tổng nhỏ nhất dương trong tất cả các mảng con của `nums` với kích thước từ `l` đến `r`. Kỹ thuật cửa sổ trượt (*sliding window*) được sử dụng để tính tổng của các mảng con một cách hiệu quả.

2.2 Các Bước Thực Hiện

1. **Xử Lý Các Trường Hợp Đặc Biệt:**
   - Nếu `nums` rỗng, trả về -1.
   - Nếu $l >$ `len(nums)` hoặc $r < l$, trả về -1 vì không có mảng con hợp lệ.
2. **Lặp Qua Các Kích Thước Mảng Con:**
   - Với mỗi kích thước mảng con trong khoảng $[l, r]$, tính tổng các mảng con bằng kỹ thuật cửa sổ trượt.
3. **Kỹ Thuật Cửa Sổ Trượt:**
   - Tính tổng của mảng con đầu tiên (cửa sổ đầu tiên).
   - Trượt cửa sổ bằng cách thêm phần tử mới vào tổng và loại bỏ phần tử đầu tiên của cửa sổ trước đó.
4. **Theo Dõi Tổng Dương Nhỏ Nhất:**
   - Duy trì một biến để lưu tổng nhỏ nhất dương.
   - Nếu không tìm thấy mảng con nào có tổng dương, trả về -1.

3 Thuật Toán

1. Khởi tạo các biến:
   - `min_sum = inf`: Lưu tổng nhỏ nhất dương tìm được.
   - `found = False`: Theo dõi liệu có mảng con hợp lệ nào được tìm thấy.
2. Lặp qua tất cả các kích thước `window_size` trong $[l, r]$.
   - Bỏ qua nếu `window_size > len(nums)`.
   - Tính tổng của cửa sổ đầu tiên.
   - Cập nhật `min_sum` và `found` nếu tổng này dương.
3. Sử dụng kỹ thuật cửa sổ trượt để tính tổng cho các cửa sổ tiếp theo:
   - Với mỗi cửa sổ mới, cập nhật tổng bằng cách thêm phần tử mới và loại bỏ phần tử đầu tiên của cửa sổ cũ.
   - Cập nhật `min_sum` và `found` nếu tìm thấy tổng dương.
4. Trả về `min_sum` nếu có mảng con hợp lệ, ngược lại trả về -1.

4 Cài Đặt Mã Lệnh

Listing 1: Python Implementation of Minimum Positive Sum Subarray

```
1
2  class Solution(object):
3      def minimumSumSubarray(self, nums, l, r):
4          """
5          :type nums: List[int]
6          :type l: int
7          :type r: int
8          :rtype: int
9          """
10         # Handle edge cases for invalid input
11         if not nums or l > len(nums) or r < l:
12             return -1
13
14         n = len(nums)
15         # Initialize the minimum sum
16         min_sum = float('inf')
17         # To track if a valid subarray is found
18         found = False
19
20         # Loop through all possible sizes from l to r
21         for window_size in range(l, r + 1):
22             # Skip if the size exceeds the array length
23             if window_size > n:
24                 break
25
26             # Compute the first window sum efficiently
27             current_sum = sum(nums[:window_size])
28
29             # If the first window satisfies the condition
30             if current_sum > 0:
31                 min_sum = min(min_sum, current_sum)
32                 found = True
33
34             # Use sliding window technique for the rest
35             for i in range(window_size, n):
36                 current_sum += nums[i] - nums[i -
                       window_size]
37                 if current_sum > 0:
38                     min_sum = min(min_sum, current_sum)
39                     found = True
40
41         # Return the result
42         return min_sum if found else -1
```

5 Phân Tích Độ Phức Tạp

5.1 Độ Phức Tạp Thời Gian

- Vòng lặp ngoài duyệt qua tất cả các kích thước cửa sổ trong $[l, r]$: $O(r - l + 1)$.
- Với mỗi kích thước cửa sổ, kỹ thuật cửa sổ trượt xử lý tất cả các phần tử của `nums`: $O(n)$.
- Tổng độ phức tạp thời gian: $O((r - l + 1) \cdot n)$.

5.2 Độ Phức Tạp Bộ Nhớ

- Thuật toán chỉ sử dụng một vài biến tạm thời, không cần cấu trúc dữ liệu bổ sung: $O(1)$.

6 Kết Luận

Thuật toán sử dụng kỹ thuật cửa sổ trượt giúp tính toán tổng của các mảng con một cách hiệu quả. Đây là giải pháp tối ưu cho kích thước đầu vào vừa phải, nhưng cần tối ưu hóa thêm nếu khoảng $[l, r]$ quá lớn.