2 Pointers Method – Phương Pháp 2 Con Trỏ

Nguyễn Quản Bá Hồng*

Ngày 9 tháng 8 năm 2025

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- *2 Pointers Method – Phương Pháp 2 Con Trỏ.*
  PDF: URL: .pdf.
  TEX: URL: .tex.
- .
  PDF: URL: .pdf.
  TEX: URL: .tex.

Mục lục

1 Introduction to 2 Pointers Method – Giới Thiệu Phương Pháp 2 Con Trỏ

Resources – Tài nguyên.

1. Antti Laaksonen. *Competitive Programmer's Handbook*. Chap. 8: Amortized Analysis. Sect. 8.1: 2 pointers method, p. 77.
2. Darren Yao, Qi Wang, Ryan Chou. USACO Guide/2 pointers. Iterating 2 monotonic pointers across an array to search for a pair of indices satisfying some condition in linear time. – Lặp lại 2 con trỏ đơn điệu trên 1 mảng để tìm kiếm 1 cặp chỉ số thỏa mãn 1 số điều kiện trong thời gian tuyến tính.
3. Geeks for Geeks/subarray with given sum.

General ideas of 2 pointers method. In the 2 pointer method, 2 pointers are used to iterate through the array values. Both pointers can move to 1 direction only, which ensures that the algorithm works efficiently.

– Trong phương pháp 2 con trỏ, 2 con trỏ được sử dụng để lặp qua các giá trị mảng. Cả hai con trỏ chỉ có thể di chuyển theo 1 hướng, điều này đảm bảo thuật toán hoạt động hiệu quả.

Cho 1 mảng $a = \{a[i]\}_{i=0}^{n-1}$. Gọi l, r lần lượt là con trỏ trái & con trỏ phải (left- & right pointers, respectively).

Problem 1 (Subarray sum). *Given an array $\{a_i\}_{i=1}^n \subset \mathbb{N}^\star$ (1-based indexing) of n positive integers & a target sum $x \in \mathbb{N}^\star$. Find a subarray whose sum is x or report that there is no such subarray.*

– Cho một mảng $\{a_i\}_{i=1}^n \subset \mathbb{N}^\star$ (đánh chỉ số dựa trên 1) gồm n số nguyên dương & một tổng đích x. Tìm một mảng con có tổng là x hoặc báo cáo rằng không có mảng con nào như vậy.

We can reformulate the problem mathematically as follows:

$$\text{Find } l, r \in [n],\ l \le r \text{ s.t. } \sum_{i=l}^{r} a_i = x.$$

Solution. This problem can be solved in $O(n)$ time (i.e., linear time) by using the 2 pointers method. The idea is to maintain pointers that point to the 1st & last value of a subarray. On each turn, the left pointer moves 1 step to the right, & the right pointer moves to the right as long as the resulting subarray sum is at most x. If the sum becomes exactly x, a solution has been found.

*A scientist- & creative artist wannabe, a mathematics & computer science lecturer of Department of Artificial Intelligence & Data Science (AIDS), School of Technology (SOT), UMT Trường Đại học Quản lý & Công nghệ TP.HCM, Hồ Chí Minh City, Việt Nam.
E-mail: nguyenquanbahong@gmail.com & hong.nguyenquanba@umt.edu.vn. Website: https://nqbh.github.io/. GitHub: https://github.com/NQBH.

– Bài toán này có thể được giải quyết trong thời gian $O(n)$ (tức là thời gian tuyến tính) bằng cách sử dụng phương pháp 2 con trỏ. Ý tưởng là duy trì các con trỏ trỏ đến giá trị đầu tiên & cuối cùng của một mảng con. Mỗi lần dịch chuyển, con trỏ trái di chuyển 1 bước sang phải, & con trỏ phải di chuyển sang phải miễn là tổng mảng con thu được không quá x. Nếu tổng bằng đúng x, thì đã tìm được nghiệm.

C++:

1. NQBH's C++: subarray sum:

```
#include <iostream>
using namespace std;

int main() {
    int n, x;
    cin >> n >> x;
    int a[n];
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
        if (a[i] == x) {
            cout << i << ' ' << i;
            return 0;
        }
    }
    int l = 0, r = 1; // left- & right pointers
    int sum = a[0] + a[1];
    while (sum != x) {
        if (sum < x && r == n - 1) {
            cout << "Impossible";
            return 0;
        }
        if (sum < x && r < n - 1) sum += a[++r];
        if (sum > x && l < n - 1) sum -= a[l++];
    }
    cout << l << ' ' << r;
}
```

□

A little bit different version of the above problem:

Problem 2. *Given a* 1*-based indexing array* `arr[]` *of nonnegative integers & an integer* x*. Return the left & right indexes (*1*-based indexing) of a subarray whose sum of its elements is equal to* x*. In case of multiple satisfying subarrays, return the subarray indexes which come 1st on moving from left to right. If no such subarray exists, return* -1*.*

2 Miscellaneous