Data Structures & Algorithms

Prefix Sum

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

Introduction

- Prefix sum is a technique that can be used on arrays (of numbers).
- The idea is to create an array prefix where prefix[i] is the sum of all elements up to the index i (inclusive).
- For example, given nums = [5, 2, 1, 6, 3, 8], we would have prefix = [5, 7, 8, 14, 17, 25].
- When a subarray starts at index 0, it is considered a "prefix" of the array. A prefix sum represents the sum of all prefixes.

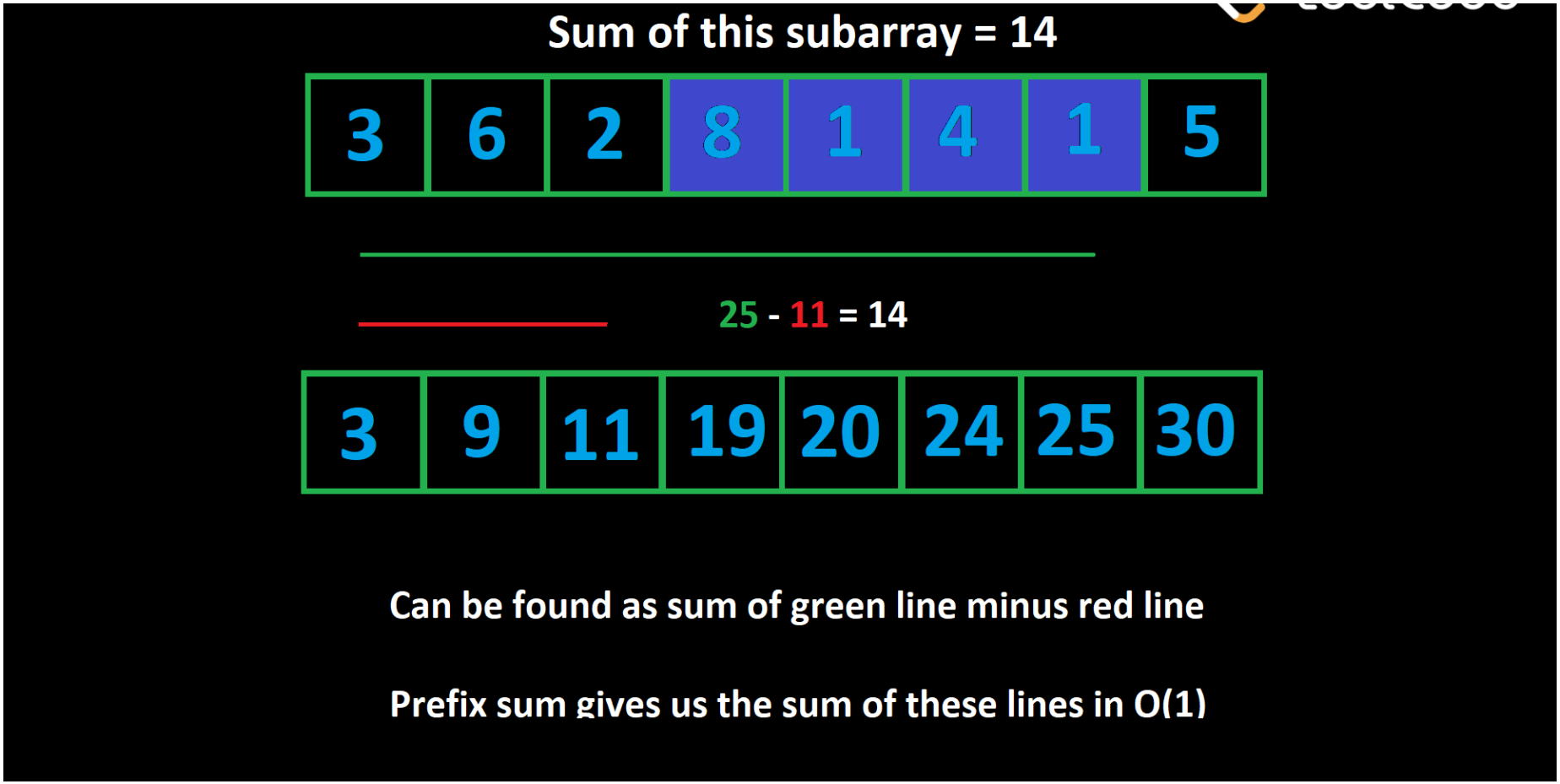
Introduction

- Prefix sums allow us to find the sum of any subarray in O(1).
- If we want the sum of the subarray from i to j (inclusive), then the answer is prefix[j] - prefix[i - 1], or prefix[j] - prefix[i] + nums[i] if you don't want to deal with the out of bounds case when i = 0.

Introduction

- This works because prefix[i - 1] is the sum of all elements before index i.
- When you subtract this from the sum of all elements up to index j, you are left with the sum of all elements starting at index i and ending at index j, which is exactly what we are looking for.

Introduction



Explanation

- In the above image, we want to find the sum of the subarray highlighted in blue.
- If you take all the elements up until the end of the subarray (the green line) and subtract all the elements before it (the red line), you have the subarray.
- With a prefix sum, we can find the sum of the green line 25 and red line 11 in constant time and take their difference to find the sum of the subarray as 14.

Pseudocode

- Given an array nums,

```
prefix = [nums[0]]
for (int i = 1; i < nums.length; i++)
    prefix.append(nums[i] + prefix[prefix.length - 1])
```

Explanation

- Initially, we start with just the first element. Then we iterate with i starting from index 1.
- At any given point, the last element of prefix will represent the sum of all the elements in the input up to but not including index i.
- So we can add that value plus the current value to the end of prefix and continue to the next element.

Explanation

- A prefix sum is a great tool whenever a problem involves sums of a subarray.
- It only costs $O(n)$ to build but allows all future subarray queries to be $O(1)$, so it can usually improve an algorithm's time complexity by a factor of $O(n)$, where n is the length of the array.

Example 1

- Given an integer array nums, an array queries where queries[i] = [x, y] and an integer limit, return a boolean array that represents the answer to each query.
- A query is true if the sum of the subarray from x to y is less than limit, or false otherwise.

Example

- For example, given nums = [1, 6, 3, 2, 7, 2], queries = [[0, 3], [2, 5], [2, 4]], and limit = 13, the answer is [true, false, true].
- For each query, the subarray sums are [12, 14, 12].

Number of Ways to Split Array

- Given an integer array nums, find the number of ways to split the array into two parts so that the first section has a sum greater than or equal to the sum of the second section.
- The second section should have at least one number.

Example

- Input
  - 10, 4, -8, 7
- Output
  - 2

Do we need the array?

- In this problem, the order in which we need to access prefix is incremental: to find leftSection, we do prefix[i] as i increments by 1 each iteration.
- As such, to calculate leftSection we don't actually need the array. We can just initialize leftSection = 0 and then calculate it on the fly by adding the current element to it at each iteration.

Do we need the array?

- What about rightSection?
  - By definition, the right section contains all the numbers in the array that aren't in the left section. Therefore, we can pre-compute the sum of the entire input as total, then calculate rightSection as total - leftSection.
- We are still using the concept of a prefix sum as each value of leftSection represents the sum of a prefix. We have simply replicated the functionality using an integer instead of an array.

Example

- Input
  - 10 4 8 3
- Output
  - 15 1 11 22

Example

- Input
  - [1,4,2,5,3]
- Output
  - 58

Queries?

Thank You…!