Newton School of Technology

1

1D List + Problem Solving Techniques

Let's do a quick recall :

- List Definition and Creation
- List Indexing: Positive and Negative
- List Slicing
- Modifying Elements in Lists
- Adding Elements in Lists
- Removing Elements in Lists
- Iterating Through Lists
- Taking direct input of lists with split()
- List Operations
- List Built In Functions and methods
- List Comprehension

Problem 1 :

Given a list of numbers and two index l and r find the sum of numbers having indexes from l to r (both inclusive) where:

Length of list <= 10^5

0<= l <= r <= (Length of list -1)

Do you think the last two lines are important?

Will the previous solution work?

Given a list of numbers and two index l and r find the sum of numbers having indexes from l to r (both inclusive) where:

Length of list <= 10^5

0<= l , r <= 10^5

SOLUTION:

You will need to handle two cases now:

1. l > r
2. l or r >= Length of list

Problem 2 :

Newton School of Technology

Given a list of numbers and q queries where each query has two indices l and r and you need to output the sum of numbers having indexes from l to r (both inclusive) for each query in a new line.

Constraints:

- **Length of list <= 10^5**
- **1<= Q <= 10^3**
- **0<= l <= r <= (Length of list -1) for each query**

Solution:

```python
def range_sum_brute_force(nums, queries):
    results = []
    for l, r in queries:
        current_sum = 0
        for i in range(l, r + 1): # Inner loop
            current_sum += nums[i]
        results.append(current_sum)
    return results
```

What is the maximum number of time the addition operator can be used in a single test case?

Question?

What is the maximum number of time the addition operator can be used in a single test case?

Max number of queries * Max No. of additions (one query)

$Q * N = (10^3 - 1) * 10^5 \sim= 10^8$

Problem 3 :

Newton School of Technology

Given a list of numbers and q queries where each query has two indices l and r and you need to output the sum of numbers having indexes from l to r (both inclusive) for each query in a new line.

Constraints:

- Length of list(N) <= 10^5
- 1<= Q <= 10^5
- 0<= l <= r <= (Length of list -1) for each query

Question?

Newton School of Technology

If we use the same solution as before what is the maximum number of time the addition operator can be used in a single test case?

No. of additions (one query) * Max number of queries

$N*Q = (10^5 - 1) * 10^5 \sim= 10^{10}$

How much time will 10^{10} operations take ?

Newton School of Technology

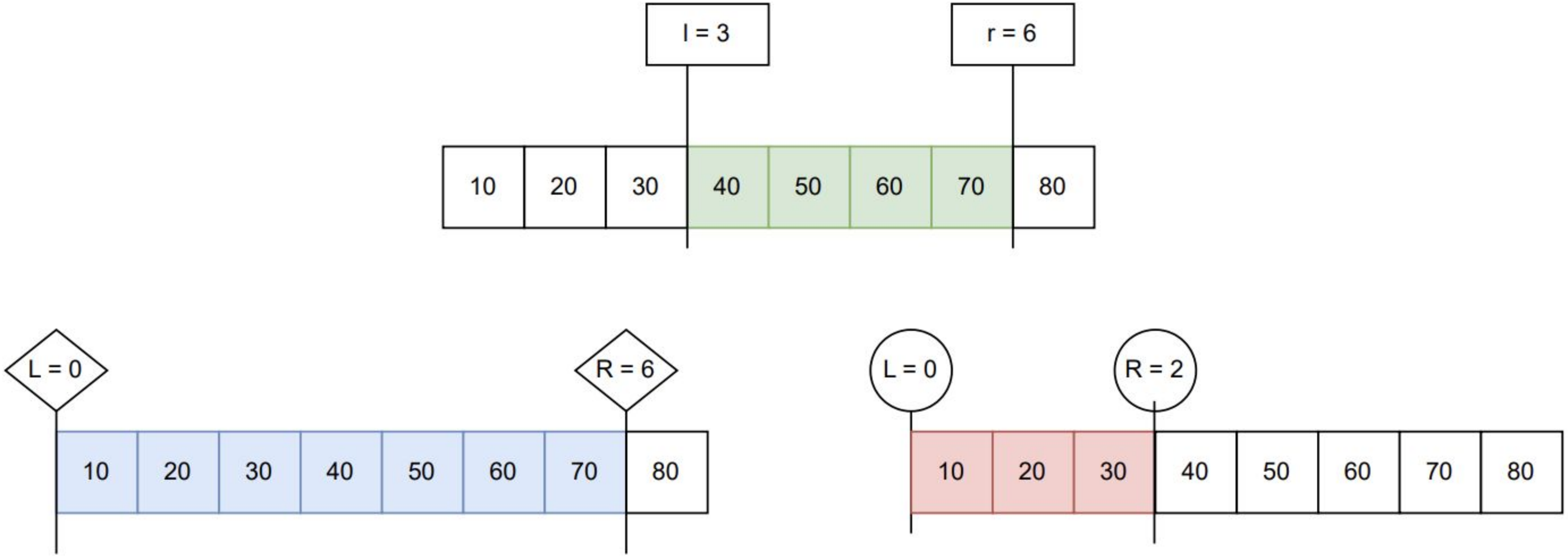
Generally we can execute 10^8 to $5*10^8$ operations in one second. For calculation purposes we should assume 10^8 only.

So time taken for 10^{10} operations will be between 20 seconds (best case) to 100 seconds (worst case).

Can we do better?

Can we do better?

Newton School of Technology



Sum(3, 6) = Sum (0, 6) - Sum(0, 2)

Can we do better?

If we have sum from 0 to l-1 and 0 to r, can we find sum from l to r?

Sum(l, r) = Sum (0, r) - Sum(0, l-1)

Now, can we use this to optimize our algorithm?

Key Idea

For an array nums of size n, construct a **prefix sum array** prefix, where:

$$\mathrm{prefix}[i] = \mathrm{nums}[0] + \mathrm{nums}[1] + \cdots + \mathrm{nums}[i]$$

Using this array, the sum of elements between indices l and r (inclusive) can be calculated in $O(1)$ time as:

$$\mathrm{sum}(l, r) = \mathrm{prefix}[r] - \mathrm{prefix}[l-1]$$

(If $l = 0$, then $\mathrm{sum}(l, r) = \mathrm{prefix}[r]$).

Prefix sum technique

It involves constructing prefix sum array where each element at index i , stores the sum of all elements in the original array from the start up to index i.

Steps to Apply Prefix Sum Technique

1. **Precompute the Prefix Sum Array:**
   - Initialize $\text{prefix}[0] = \text{nums}[0]$.
   - For $i \geq 1$, compute $\text{prefix}[i] = \text{prefix}[i-1] + \text{nums}[i]$.
2. **Answer Queries:**
   - For a query (l, r), calculate $\text{prefix}[r] - \text{prefix}[l-1]$.

Prefix sum dry run

Let’s say query has l = 3, r = 6

Given List

10	20	30	40	50	60	70	80

Prefix Sum List

10	30	60	100	150	210	280	360

l = 3

r = 6

Prefix sum dry run

Let’s say query has l = 3, r = 6

10	20	30	40	50	60	70	80

prefix_sum[r] = prefix_sum[6] = 280

10	20	30	40	50	60	70	80

prefix_sum[l-1] = prefix_sum[2] = 60

10	20	30	40	50	60	70	80

prefixSum[6] - prefixSum[2] = 280 - 60 = 220

10	20	30	40	50	60	70	80

Optimal Code: Prefix sum

Newton School of Technology

```python
def range_sum_prefix(nums, queries):
    # Step 1: Precompute the prefix sum array
    n = len(nums)
    prefix = [0] * n
    prefix[0] = nums[0]

    for i in range(1, n):
        prefix[i] = prefix[i - 1] + nums[i]

    # Step 2: Answer each query in O(1)
    results = []
    for l, r in queries:
        if l == 0:
            results.append(prefix[r])
        else:
            results.append(prefix[r] - prefix[l - 1])

    return results
```

Maximum number of operations:

Newton School of Technology

- Operations required to construct prefix sum array = $c_1 * n$
- Operations required to answer one query = k
- Operations required to answer q queries = $k * q$

Total number of operations = $c_1 * n + q * k$

Thank You!