#### Chandigarh University Logo: Overview - CareerGuide

#### STACK AND QUEUE STANDARD QUESTION

#### VERY EASY:

#### 1 Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

#### Implement the MinStack class:

#### MinStack() initializes the stack object.

#### void push(int val) pushes the element val onto the stack.

#### void pop() removes the element on the top of the stack.

#### int top() gets the top element of the stack.

#### int getMin() retrieves the minimum element in the stack.

#### You must implement a solution with O(1) time complexity for each function.

#### Example 1:

#### Input

#### ["MinStack","push","push","push","getMin","pop","top","getMin"]

#### [[],[-2],[0],[-3],[],[],[],[]]

#### Output

#### [null,null,null,null,-3,null,0,-2]

#### Explanation

#### MinStack minStack = new MinStack();

#### minStack.push(-2);

#### minStack.push(0);

#### minStack.push(-3);

#### minStack.getMin(); // return -3

#### minStack.pop();

#### minStack.top(); // return 0

#### minStack.getMin(); // return -2

#### Example 2:

**Input:**

["MinStack", "push", "push", "push", "push", "getMin", "pop", "getMin", "top", "getMin"]

[[], [5], [3], [7], [3], [], [], [], [], []]

**Output**

[null, null, null, null, null, 3, null, 3, 7, 3]

**Explanation:**

MinStack minStack = new MinStack();

minStack.push(5); # Stack: [5], MinStack: [5]

minStack.push(3); # Stack: [5, 3], MinStack: [5, 3]

minStack.push(7); # Stack: [5, 3, 7], MinStack: [5, 3]

minStack.push(3); # Stack: [5, 3, 7, 3], MinStack: [5, 3, 3]

minStack.getMin(); # Returns 3

minStack.pop(); # Removes 3; Stack: [5, 3, 7], MinStack: [5, 3]

minStack.getMin(); # Returns 3

minStack.top(); # Returns 7

minStack.getMin(); # Returns 3

* Minimum values are maintained as: [5] → [5, 3] → [5, 3] → [5, 3]
* After pops, the minimum values update accordingly.

#### Example 3:

**Input:**

["MinStack", "push", "push", "push", "getMin", "pop", "getMin", "pop", "getMin"]

[[], [2], [1], [4], [], [], [], [], []]

**Output:**

[null, null, null, null, 1, null, 1, null, 2]

**Explanation:**

minStack = MinStack()

minStack.push(2)

minStack.push(1)

minStack.push(4)

minStack.push(1)

print(minStack.getMin()) # Output: 1

minStack.pop()

print(minStack.getMin()) # Output: 1

minStack.pop()

print(minStack.getMin()) # Output: 1

minStack.pop()

print(minStack.getMin()) # Output: 2

* Minimum values are maintained as: [2] → [2, 1] → [2, 1] → [2, 1]
* After pops, the minimum values update accordingly.

#### Constraints:

#### -2^31 <= val <= 2^31 - 1

#### Methods pop, top and getMin operations will always be called on non-empty stacks.

#### At most 3 \* 10^4 calls will be made to push, pop, top, and getMin. Ø Sources:<https://leetcode.com/problems/min-stack/>

**2 Given a string s, find the first non-repeating character in it and return its index. If it does not exist, return -1.**

**Example 1:**

**Input:** s = "leetcode"

**Output:** 0

**Explanation:**

The character 'l' at index 0 is the first character that does not occur at any other index.

**Example 2:**

**Input:** s = "loveleetcode"

**Output:** 2

**Example 3:**

**Input:** s = "aabb"

**Output:** -1

**Constraints:**

* 1 <= s.length <= 105
* s consists of only lowercase English letters.

**Approach:**

Use a hash map or array of size 26 to store the frequency of each character (since the input consists of lowercase English letters).

Traverse the string to count the frequency of each character.

Traverse the string again to find the first character with a frequency of 1. Return its index.

If no such character is found, return -1.

**Time Complexity:** ( O(n) ), where ( n ) is the length of the string. The string is traversed twice.

**Space Complexity:** ( O(1) ), as the frequency array has a fixed size of 26.

Reference : <https://leetcode.com/problems/first-unique-character-in-a->

[string/description/](https://leetcode.com/problems/first-unique-character-in-a-)

#### 3 Implement a simple text editor. The editor initially contains an empty string, S.Perform Q  operations of the following 4 types:

#### append(W) - Append string W to the end of S.

#### delete (k)- Delete the last k characters of S.

#### print (k)- Print the k^th  character of  S.

#### undo() - Undo the last (not previously undone) operation of type 1 or 2, reverting  S to the state it was in prior to that operation.

#### Example 1

#### S = ‘abcde’

#### Ops=[‘1 fg’, ‘3 6’, ‘2 5’, ‘4’, ‘3 7’, ‘4’, ‘3 4’]

#### operation

#### index S ops[index] explanation

#### ----- ------ ---------- -----------

#### 0 abcde 1 fg append fg

#### 1 abcdefg 3 6 print the 6th letter - f

#### 2 abcdefg 2 5 delete the last 5 letters

#### 3 ab 4 undo the last operation, index 2

#### 4 abcdefg 3 7 print the 7th characgter - g

#### 5 abcdefg 4 undo the last operation, index 0

#### 6 abcde 3 4 print the 4th character – d

#### The results should be printed as:

#### f

#### g

#### d

#### Input Format

#### The first line contains an integer,Q , denoting the number of operations. Each line i of the Q  subsequent lines (where 0≤i<Q ) defines an operation to be performed. Each operation starts with a single integer, t (where t  ∈ {1,2,3,4}), denoting a type of operation as defined in the Problem Statement above. If the operation requires an argument, t is followed by its space-separated argument. For example, if t=1 and W=”abcd” , line i will be 1 abcd.

### Example 2 (Custom Test Case)

#### Input

#### 9

1 hello

1 world

3 10

2 5

3 5

4

3 10

4

3 1

#### Code Execution

operations = [

"1 hello", "1 world", "3 10", "2 5", "3 5",

"4", "3 10", "4", "3 1"

]

result = text\_editor(operations)

print("\n".join(result)) # Outputs: d, o, d, h

#### Output

d

o

d

h

### Example 3 (Custom Test Case)

#### Input

10

1 programming

3 1

2 6

3 4

1 code

3 4

4

3 7

4

3 8

#### Code Execution

operations = [

"1 programming", "3 1", "2 6", "3 4", "1 code",

"3 4", "4", "3 7", "4", "3 8"

]

result = text\_editor(operations)

print("\n".join(result)) # Outputs: p, g, o, m

#### Output

p

g

o

m

#### Constraints

#### 1≤ Q ≤ 106

#### 1 ≤ k ≤ |S|

#### The sum of the lengths of all  in the input ≤ 106 .

#### The sum of  over all delete operations ≤ 2.106 .

#### All input characters are lowercase English letters.

#### It is guaranteed that the sequence of operations given as input is possible to perform.

#### Output Format

#### Each operation of type 3 must print the k^th  character on a new line.

#### Sample Input

#### STDIN Function

#### ----- --------

#### 8 Q = 8

#### 1 abc ops[0] = '1 abc'

#### 3 3 ops[1] = '3 3'

#### 2 3 ...

#### 1 xy

#### 3 2

#### 4

#### 4

#### 3 1

#### Sample Output

#### c

#### y

#### a

#### Explanation

#### Initially,S  is empty. The following sequence of  8 operations are described below:

#### S=””. We append abc to S , so S = “abc”.

#### Print the 3rdcharacter on a new line. Currently, the  3rd character is c.

#### Delete the last  3 characters in  S(abc), so S= “”.

#### Append  xy to S , so S= “xy”.

#### Print the 2ndcharacter on a new line. Currently, the 2nd  character is y.

#### Undo the last update to S, making  S empty again (i.e.,S=”” ).

#### Undo the next to last update to S (the deletion of the last  3 characters), making S=”abc”.

#### Print the 1st character on a new line. Currently, the  1st character is a.

* **Frequently Asked By**: Facebook, Atlassian, Adobe(**Year**: 2018–2022)
* Reference: <https://www.hackerrank.com/challenges/simple-text-editor>

**4** **Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).**

**Implement the MyQueue class:**

**void push(int x) Pushes element x to the back of the queue.**

**int pop() Removes the element from the front of the queue and returns it.**

**int peek() Returns the element at the front of the queue.**

**boolean empty() Returns true if the queue is empty, false otherwise.**

**Notes:**

You must use only standard operations of a stack, which means only push to top, peek/pop from top, size, and is empty operations are valid.

Depending on your language, the stack may not be supported natively. You may simulate a stack using a list or deque (double-ended queue) as long as you use only a stack's standard operations.

**Example 1:**

**Input**

["MyQueue", "push", "push", "peek", "pop", "empty"]

[[], [1], [2], [], [], []]

**Output**

[null, null, null, 1, 1, false]

**Explanation**

* MyQueue myQueue = new MyQueue();
* myQueue.push(1); // queue is: [1]
* myQueue.push(2); // queue is: [1, 2] (leftmost is front of the queue)
* myQueue.peek(); // return 1
* myQueue.pop(); // return 1, queue is [2]
* myQueue.empty(); // return false

**Example 2:**

**Input:**

["MyQueue", "push", "push", "push", "peek", "pop", "peek", "empty"]

[[], [3], [5], [7], [], [], [], []]

**Output:**

[null, null, null, null, 3, 3, 5, false]

**Explanation:**

MyQueue myQueue = new MyQueue()

myQueue.push(3) # queue is: [3]

myQueue.push(5) # queue is: [3, 5]

myQueue.push(7) # queue is: [3, 5, 7]

myQueue.peek() # return 3

myQueue.pop() # return 3, queue is: [5, 7]

myQueue.peek() # return 5

myQueue.empty() # return False

**Example 3:**

**Input:**

["MyQueue", "push", "peek", "push", "pop", "pop", "empty"]

[[], [8], [], [10], [], [], []]

**Output:**

[null, null, 8, null, 8, 10, true]

**Explanation:**

MyQueue myQueue = new MyQueue()

myQueue.push(8) # queue is: [8]

myQueue.peek() # return 8

myQueue.push(10) # queue is: [8, 10]

myQueue.pop() # return 8, queue is: [10]

myQueue.pop() # return 10, queue is: []

myQueue.empty() # return True

**Constraints:**

* 1 <= x <= 9
* At most 100 calls will be made to push, pop, peek, and empty.
* All the calls to pop and peek are valid.

**Asked By :** Amazon,Google

**Reference** :<https://leetcode.com/problems/implement-queue-using-stacks/description/>

#### 5 You are given an array of strings tokens that represents an arithmetic expression in a Reverse Polish Notation.

#### Evaluate the expression. Return an integer that represents the value of the expression.

#### Note that:

#### The valid operators are '+', '-', '\*', and '/'.

#### Each operand may be an integer or another expression.

#### The division between two integers always truncates toward zero.

#### There will not be any division by zero.

#### The input represents a valid arithmetic expression in a reverse polish notation.

#### The answer and all the intermediate calculations can be represented in a 32-bit integer.

#### Example 1:

#### Input: tokens = ["2","1","+","3","\*"]

#### Output: 9

#### Explanation: ((2 + 1) \* 3) = 9

#### Example 2:

#### Input: tokens = ["4","13","5","/","+"]

#### Output: 6

#### Explanation: (4 + (13 / 5)) = 6

#### Example 3:

#### Input: tokens = ["10","6","9","3","+","-11","\*","/","\*","17","+","5","+"]

#### Output: 22

#### Explanation: ((10 \* (6 / ((9 + 3) \* -11))) + 17) + 5

#### = ((10 \* (6 / (12 \* -11))) + 17) + 5

#### = ((10 \* (6 / -132)) + 17) + 5

#### = ((10 \* 0) + 17) + 5

#### = (0 + 17) + 5

#### = 17 + 5

#### = 22

#### Constraints:

#### 1 <= tokens.length <= 104

#### tokens[i] is either an operator: "+", "-", "\*", or "/", or an integer in the range [-200, 200].

#### Approach

#### Using Fundamentals of STACK && LAMBDA...

#### Time complexity: O(n)

#### Space complexity: O(n)

**Sources**[: https://leetcode.com/problems/evaluate-reverse-polish-notation/](https://leetcode.com/problems/evaluate-reverse-polish-notation/)

**Frequently Asked By**: Amazon, Google, Apple(**Year**: 2019–2023)

#### EASY LEVEL:-

#### 6 A bracket is considered to be any one of the following characters: (, ), {, }, [, or ].

#### Two brackets are considered to be a matched pair if the an opening bracket (i.e., (, [, or {) occurs to the left of a closing bracket (i.e., ), ], or }) of the exact same type. There are three types of matched pairs of brackets: [], {}, and ().

#### A matching pair of brackets is not balanced if the set of brackets it encloses are not matched. For example, {[(])} is not balanced because the contents in between { and } are not balanced. The pair of square brackets encloses a single, unbalanced opening bracket, (, and the pair of parentheses encloses a single, unbalanced closing square bracket, ].

#### By this logic, we say a sequence of brackets is balanced if the following conditions are met:

#### It contains no unmatched brackets.

#### The subset of brackets enclosed within the confines of a matched pair of brackets is also a matched pair of brackets.

#### Given n  strings of brackets, determine whether each sequence of brackets is balanced. If a string is balanced, return YES. Otherwise, return NO.

#### Function Description

#### Complete the function isBalanced in the editor below.

#### isBalanced has the following parameter(s):

#### string s: a string of brackets

#### Returns

#### string: either YES or NO

#### Input Format

#### The first line contains a single integer n, the number of strings. Each of the next n lines contains a single string s, a sequence of brackets.

#### Constraints

#### 1 ≤ n ≤ 10^3

#### 1 ≤ |S| ≤ 10^3, where |S| is the length of the sequence.

#### All chracters in the sequences ∈ { {, }, (, ), [, ] }.

#### Output Format

#### For each string, return YES or NO.

### Example 1:

#### Sample Input

#### STDIN Function

#### ----- --------

#### 3 n = 3

#### {[()]} first s = '{[()]}'

#### {[(])} second s = '{[(])}'

#### {{[[(())]]}} third s ='{{[[(())]]}}'

#### Sample Output

#### YES

#### NO

#### YES

#### Explanation

#### The string {[()]} meets both criteria for being a balanced string.

#### The string {[(])} is not balanced because the brackets enclosed by the matched pair { and } are not balanced: [(]).

#### The string {{[[(())]]}} meets both criteria for being a balanced string.

### Example 2

#### Input

4

()

([{}])

(]

{[()]}

#### Output

YES

YES

NO

YES

### Example 3

#### Input

5

{}

[{}()]

[{()}]

{[}]

((({[()]})))

#### Output

YES

YES

YES

NO

YES

### Explanation of Custom Test Cases

#### Custom Test Case 1:

1. () → Balanced because it consists of a single pair of matched brackets.
2. ([{}]) → Balanced because each pair of brackets is properly matched and nested.
3. (] → Not balanced because ( is closed with ].
4. {[()]} → Balanced because all brackets are properly matched and nested.

#### Custom Test Case 2:

1. {} → Balanced because it consists of a single pair of matched brackets.
2. [{}()] → Balanced because all brackets are properly matched and nested.
3. [{()}] → Balanced because all brackets are properly matched and nested.
4. {[}] → Not balanced because [} are mismatched.
5. ((({[()]}))) → Balanced because all brackets are properly matched and nested.

#### Reference: <http://leetcode.com/problems/valid-parentheses/>

#### 7 The school cafeteria offers circular and square sandwiches at lunch break, referred to by numbers 0 and 1 respectively. All students stand in a queue. Each student either prefers square or circular sandwiches.

**The number of sandwiches in the cafeteria is equal to the number of students. The sandwiches are placed in a stack. At each step:**

If the student at the front of the queue prefers the sandwich on the top of the stack, they will take it and leave the queue.

Otherwise, they will leave it and go to the queue's end.

This continues until none of the queue students want to take the top sandwich and are thus unable to eat.

You are given two integer arrays students and sandwiches where sandwiches[i] is the type of the i​​​​​​th sandwich in the stack (i = 0 is the top of the stack) and students[j] is the preference of the j​​​​​​th student in the initial queue (j = 0 is the front of the queue). Return the number of students that are unable to eat.

**Example 1:**

**Input:** students = [1,1,0,0], sandwiches = [0,1,0,1]

**Output:** 0

**Explanation:**

- Front student leaves the top sandwich and returns to the end of the line making students = [1,0,0,1].

- Front student leaves the top sandwich and returns to the end of the line making students = [0,0,1,1].

- Front student takes the top sandwich and leaves the line making students = [0,1,1] and sandwiches = [1,0,1].

- Front student leaves the top sandwich and returns to the end of the line making students = [1,1,0].

- Front student takes the top sandwich and leaves the line making students = [1,0] and sandwiches = [0,1].

- Front student leaves the top sandwich and returns to the end of the line making students = [0,1].

- Front student takes the top sandwich and leaves the line making students = [1] and sandwiches = [1].

- Front student takes the top sandwich and leaves the line making students = [] and sandwiches = [].

Hence all students are able to eat.

**Example 2:**

**Input:** students = [1,1,1,0,0,1], sandwiches = [1,0,0,0,1,1]

**Output:** 3

**Constraints:**

* 1 <= students.length, sandwiches.length <= 100
* students.length == sandwiches.length
* sandwiches[i] is 0 or 1.
* students[i] is 0 or 1.

**Approach**

* Create two queues of students and sandwiches
* And a count variable to check if is loop in left student
* If students in the queue cannot have their ordered sandwiches, it makes a loop. If it is a loop, just break and return the result
* Then implement the program like the given rules.

**Time complexity:** O(n)

**Space complexity:** O(n)

**Asked By: Tcs,Optum**

**Reference** <https://leetcode.com/problems/number-of-students-unable-to-eat-lunch/description/>

#### 8 We are given an array asteroids of integers representing asteroids in a row.

#### For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed.

#### Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are the same size, both will explode. Two asteroids moving in the same direction will never meet.

#### Example 1:

#### Input: asteroids = [5,10,-5]

#### Output: [5,10]

#### Explanation: The 10 and -5 collide resulting in 10. The 5 and 10 never collide.

#### Example 2:

#### Input: asteroids = [8,-8]

#### Output: []

#### Explanation: The 8 and -8 collide exploding each other.

#### Example 3:

#### Input: asteroids = [10,2,-5]

#### Output: [10]

#### Explanation: The 2 and -5 collide resulting in -5. The 10 and -5 collide resulting in 10.

#### Constraints:

#### 2 <= asteroids.length <= 104

#### -1000 <= asteroids[i] <= 1000

#### asteroids[i] != 0

#### Approach

#### Start by iterating through the array of asteroids while using a stack to keep track of the remaining asteroids.

#### For each asteroid:

#### If the stack isn't empty and the current asteroid moves left (negative) while the stack's top asteroid moves right (positive), check for collisions.

#### In a collision:

#### If the current asteroid is larger in size, pop the stack (indicating the top asteroid is destroyed) and continue comparing until no more collisions can occur.

#### If both asteroids are of equal size, pop the stack and ignore the current asteroid (both are destroyed).

#### If the top asteroid is larger, don't push the current asteroid onto the stack.

#### If no collision occurs, or the stack is empty, push the current asteroid onto the stack.

#### Once all asteroids have been processed, the stack will contain the remaining asteroids in order.

#### Reverse the stack to obtain the correct sequence, as the last asteroid pushed should appear last in the result.

#### Complexity

#### Time complexity: O(N), since each asteroid is processed at most once.

#### Space complexity: O(N), to account for the stack storage in the worst case where no collisions occur.

* **Frequently Asked By**: Microsoft, Amazon

#### **Year**: 2020–2023

* **Sources**:<https://leetcode.com/problems/asteroid-collision/>

**9 Given an integer array nums, handle multiple queries of the following type:**

**Calculate the sum of the elements of nums between indices left and right inclusive where left <= right.**

**Implement the NumArray class:**

**NumArray(int[] nums) Initializes the object with the integer array nums.**

**int sumRange(int left, int right) Returns the sum of the elements of nums between indices left and right inclusive (i.e. nums[left] + nums[left + 1] + ... + nums[right]).**

**Example 1:**

**Input :**["NumArray", "sumRange", "sumRange", "sumRange"]

[[[-2, 0, 3, -5, 2, -1]], [0, 2], [2, 5], [0, 5]]

**Output:**[null, 1, -1, -3]

**Explanation**

NumArray numArray = new NumArray([-2, 0, 3, -5, 2, -1]);

numArray.sumRange(0, 2); // return (-2) + 0 + 3 = 1

numArray.sumRange(2, 5); // return 3 + (-5) + 2 + (-1) = -1

numArray.sumRange(0, 5); // return (-2) + 0 + 3 + (-5) + 2 + (-1) = -3

**Constraints:**

* 1 <= nums.length <= 104
* -105 <= nums[i] <= 105
* 0 <= left <= right < nums.length
* At most 104 calls will be made to sumRange.

**Approach**

Let's Say Array Is : [ -2, 0, 3, -5, 2, -1 ]

I Create Prefix Sum Array A: [ 0, -2, -2, 1, -4, -2, -3 ]

How ?

Prefix Sum For Index i : Sum Of All Elements Before i

So For Sum from l to r : A[r+1] - A[l]

A[r+1] Sum of All Element from `0 To r`

A[l] Sum of All Element from `0 To l-1`

**Asked By: Microsoft**

#### Reference :<https://leetcode.com/problems/range-sum-query-immutable/description/>

#### 10 You have an empty sequence, and you will be given  N queries. Each query is one of these three types:

#### x -Push the element x into the stack.

#### Delete the element present at the top of the stack.

#### Print the maximum element in the stack.

#### **Function Description**

#### Complete the getMax function in the editor below.

#### getMax has the following parameters: - string operations[n]: operations as strings

#### **Returns** - int[]: the answers to each type 3 query

#### **Input Format**

#### The first line of input contains an integer, n. The next  n lines each contain an above mentioned query.

#### **Constraints**

#### 1 ≤ n ≤ 10^5

#### 1 ≤ x ≤ 10^9

#### 1 ≤ type ≤ 3, All queries are valid.

#### **Sample Input**

#### STDIN Function

#### ----- --------

#### 10 operations[] size n = 10

#### 1 97 operations = ['1 97', '2', '1 20', ....]

#### 2

#### 1 20

#### 2

#### 1 26

#### 1 20

#### 2

#### 3

#### 1 91

#### 3

#### **Sample Output**

#### 26

#### 91

### Example 2

#### Input

8

1 5

1 1

1 3

3

2

3

2

3

#### Output

5

5

5

#### Explanation

1. Push 5 → max is 5.
2. Push 1 → max is still 5.
3. Push 3 → max remains 5.
4. Print max → output 5.
5. Delete 3 → max is still 5.
6. Print max → output 5.
7. Delete 1 → max remains 5.
8. Print max → output 5.

### Example 3

#### Input

7

1 10

1 20

1 30

3

2

3

2

#### Output

30

20

#### Explanation

1. Push 10 → max is 10.
2. Push 20 → max is 20.
3. Push 30 → max becomes 30.
4. Print max → output 30.
5. Delete 30 → max becomes 20.
6. Print max → output 20.
7. Delete 20 → max is now 10 (not printed as there's no query).

#### Asked By : Wipro, Optum

#### Reference :<https://www.hackerrank.com/challenges/maximum-element/problem>

#### MEDIUM:-

#### 11 Given a circular integer array nums (i.e., the next element of nums[nums.length - 1] is nums[0]), return the next greater number for every element in nums.

#### The next greater number of a number x is the first greater number to its traversing-order next in the array, which means you could search circularly to find its next greater number. If it doesn't exist, return -1 for this number.

#### Example 1:

#### Input: nums = [1,2,1]

#### Output: [2,-1,2]

#### Explanation:

#### The first 1's next greater number is 2;

#### The number 2 can't find next greater number.

#### The second 1's next greater number needs to search circularly, which is also 2.

#### Example 2:

#### Input: nums = [1,2,3,4,3]

#### Output: [2,3,4,-1,4]

#### Constraints:

#### 1 <= nums.length <= 104

#### -109 <= nums[i] <= 109

#### Asked By : Tcs Sde1,Wipro

#### Reference :<https://leetcode.com/problems/next-greater-element-ii/description/>

#### 12 Given a queue, write a recursive function to reverse it.

**Standard operations allowed :**

**enqueue(x) : Add an item x to rear of queue.   
dequeue() : Remove an item from front of queue.   
empty() : Checks if a queue is empty or not.**

**Examples 1:**

**Input : Q = [5, 24, 9, 6, 8, 4, 1, 8, 3, 6]**

**Output : Q = [6, 3, 8, 1, 4, 8, 6, 9, 24, 5]**

**Explanation : Output queue is the reverse of the input queue.**

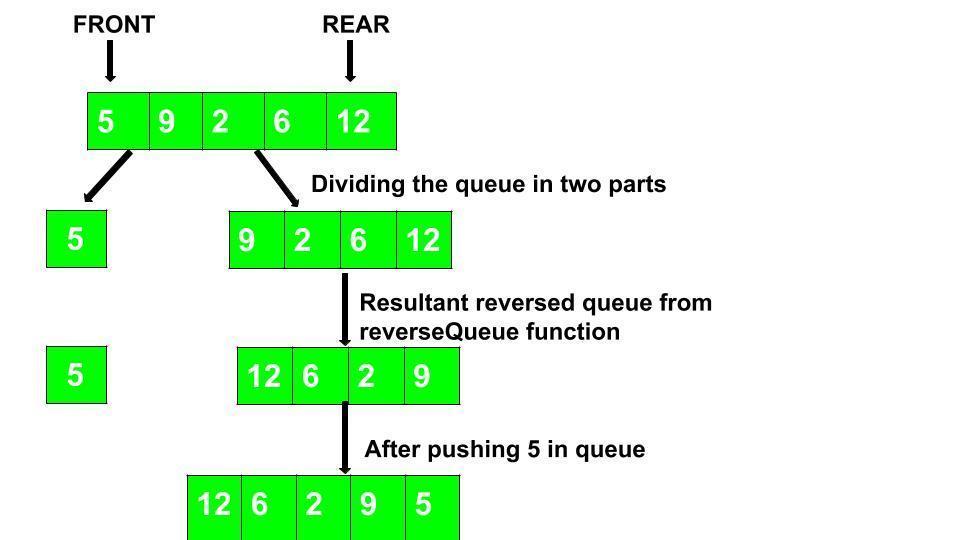
**Example 2:**

**Input : Q = [8, 7, 2, 5, 1]**

**Output : Q = [1, 5, 2, 7, 8]**

**Recursive Algorithm :**

1. The pop element from the queue if the queue has elements otherwise return empty queue.
2. Call reverseQueue function for the remaining queue.
3. Push the popped element in the resultant reversed queue.



**Pseudo Code :**

queue reverseFunction(queue)

{

if (queue is empty)

return queue;

else {

data = queue.front()

queue.pop()

queue = reverseFunction(queue);

q.push(data);

return queue;

}

}

**Asked By: Microsoft,Phonepe**

**Reference :**<https://www.geeksforgeeks.org/reversing-queue-using-recursion/>

#### 13. Given a balanced parentheses string s, return the score of the string.

#### The score of a balanced parentheses string is based on the following rule:

#### "()" has score 1.

#### AB has score A + B, where A and B are balanced parentheses strings.

#### (A) has score 2 \* A, where A is a balanced parentheses string.

#### Example 1:

#### Input: s = "()"

#### Output: 1

#### Example 2:

#### Input: s = "(())"

#### Output: 2

#### Example 3:

#### Input: s = "()()"

#### Output: 2

#### Constraints:

#### 2 <= s.length <= 50

#### s consists of only '(' and ')'.

#### s is a balanced parentheses string.

#### Approach

#### The question is simple but is designed such that certain edge cases has to be handled. Read the solution to understand, how this is done

#### Time complexity: O(n)

#### Space complexity: O(n)

#### Asked By : Tcs Sde1,Wipro

#### Reference :<https://leetcode.com/problems/score-of-parentheses/description/>

#### 14 You are given a 0-indexed string pattern of length n consisting of the characters 'I' meaning increasing and 'D' meaning decreasing.

#### A 0-indexed string num of length n + 1 is created using the following conditions:

#### num consists of the digits '1' to '9', where each digit is used at most once.

#### If pattern[i] == 'I', then num[i] < num[i + 1].

#### If pattern[i] == 'D', then num[i] > num[i + 1].

#### Return the lexicographically smallest possible string num that meets the conditions.

#### Example 1:

#### Input: pattern = "IIIDIDDD"

#### Output: "123549876"

#### Explanation:

#### At indices 0, 1, 2, and 4 we must have that num[i] < num[i+1].

#### At indices 3, 5, 6, and 7 we must have that num[i] > num[i+1].

#### Some possible values of num are "245639871", "135749862", and "123849765".

#### It can be proven that "123549876" is the smallest possible num that meets the conditions.

#### Note that "123414321" is not possible because the digit '1' is used more than once.

#### Example 2:

#### Input: pattern = "DDD"

#### Output: "4321"

#### Explanation:

#### Some possible values of num are "9876", "7321", and "8742".

#### It can be proven that "4321" is the smallest possible num that meets the conditions.

#### Constraints:

#### 1 <= pattern.length <= 8

#### pattern consists of only the letters 'I' and 'D'.

#### Asked By : Tcs Sde1,Wipro

#### Reference :<https://leetcode.com/problems/construct-smallest-number-from-di-string/description/>

15 **Given a 2D image img**[][]**where each img**[i][j]**is an integer representing the color of that pixel, also given the location of a pixel**(x, y)**and a new color**newClr**, the task is to replace the existing color of the given pixel and all the adjacent same-colored pixels with the given newClr.**

**Example 1:**

**Input**: img[][] =  
{ {1, 1, 1},   
{1, 1, 0},   
{1, 0, 1} }  
x = 1, y = 1, newClr = 3  
**Output**: img[][] =  
{{3, 3, 3},  
{3, 3, 0},   
{3, 0, 1}}  
**Explanation**: The value at arr[1][1] is 1. All connected pixels with the color 1 are replaced with 3

**Example 2:**

**Input**: img[][] =   
{ {0, 0, 0},   
{0, 1, 1} }  
x = 1, y = 1, newClr = 1  
**Output**: img[][] =  
{{0, 0, 0},  
{0, 1 , 1}}  
**Explanation**: Old and new colors are same, so no change

**Example 3:**

**Input**:: arr[][] =   
{ {2, 2, 2},   
{2, 2, 2}, }  
x = 0, y = 0, newClr = 1  
**Output**: img[][] =  
{{1, 1, 1},  
{1, 1, 1}}}

**Code :**

## Using DFS (O(m x n)) :

* Change the color of the source row and source column with the given color
* Do DFS in four directions
* Do not forget to handle the case when previous and new colors are same.

**Time Complexity:** O(m\*n)  
**Auxiliary Space:** O(m\*n), due to the recursive call stack.

Further Optimizations to the above code. We can check for the condition **if (img[x][y] != prevClr)**before making recursive calls. This will save some unnecessary recursive calls for the adjacent of the last nodes.

**Asked By: Google,Phonepe,MorganStanly**

**Reference :** [**https://www.geeksforgeeks.org/flood-fill-algorithm/**](https://www.geeksforgeeks.org/flood-fill-algorithm/)

#### HARD

#### 16 Given an array  A[] of N  distinct elements. Let  M1 and M2 be the smallest and the next smallest element in the interval [L,R]  where  1 ≤ L ≤ R ≤ N.

#### Si = (((M1^M2) ⊕ (M1 V M2)) (M1 ⊕ M2))

#### Where ^,V, ⊕ , are the bitwise operators AND,OR and XOR respectively. Your task is to find the maximum possible value of  Si.

#### Input Format

#### First line contains integer N.

#### Second line contains N  integers, representing elements of the array A[] .

#### Constraints

#### 1 ≤ N ≤ 106

#### 1 ≤ Ai ≤ 109.

#### Output Format

#### Print the value of maximum possible value of Si .

#### Sample Input

#### 5

#### 9 6 3 5 2

#### Sample Output

#### 15

#### Explanation

#### Consider the interval [1,2]  the result will be maximum.

#### (((9^6) ⊕ (9 V 6)) (9 ⊕ 6)) = 15

#### Example 2:

**Input:**

6

12 8 3 5 7 6

**Output:**

15

**Explanation:**

* The maximum value of SiSiSi is 15 when considering M1=12M1 = 12M1=12 and M2=8M2 = 8M2=8.

 **Frequently Asked By**: Facebook, Google(**Year**: 2018–2022)  
 **Refrences**:<https://www.hackerrank.com/challenges/and-xor-or/problem?isFullScreen=true>

**17 You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.**

**Return the max sliding window**.

**Example 1:**

**Input:** nums = [1,3,-1,-3,5,3,6,7], k = 3

**Output:** [3,3,5,5,6,7]

**Explanation:**

Window position Max

--------------- -----

[1 3 -1] -3 5 3 6 7 3

1 [3 -1 -3] 5 3 6 7 3

1 3 [-1 -3 5] 3 6 7 5

1 3 -1 [-3 5 3] 6 7 5

1 3 -1 -3 [5 3 6] 7 6

1 3 -1 -3 5 [3 6 7] 7

**Example 2:**

**Input:** nums = [1], k = 1

**Output:** [1]

**Constraints:**

* 1 <= nums.length <= 105
* -104 <= nums[i] <= 104
* 1 <= k <= nums.length

**Asked By : Amazon 2023**

**Referenece :**<https://leetcode.com/problems/sliding-window-maximum/description/>

#### 18 You have an infinite number of stacks arranged in a row and numbered (left to right) from 0, each of the stacks has the same maximum capacity.

#### Implement the DinnerPlates class:

#### DinnerPlates(int capacity) Initializes the object with the maximum capacity of the stacks capacity.

#### void push(int val) Pushes the given integer val into the leftmost stack with a size less than capacity.

#### int pop() Returns the value at the top of the rightmost non-empty stack and removes it from that stack, and returns -1 if all the stacks are empty.

#### int popAtStack(int index) Returns the value at the top of the stack with the given index index and removes it from that stack or returns -1 if the stack with that given index is empty.

#### Example 1:

#### Input

#### ["DinnerPlates", "push", "push", "push", "push", "push", "popAtStack", "push", "push", "popAtStack", "popAtStack", "pop", "pop", "pop", "pop", "pop"]

#### [[2], [1], [2], [3], [4], [5], [0], [20], [21], [0], [2], [], [], [], [], []]

#### Output

#### [null, null, null, null, null, null, 2, null, null, 20, 21, 5, 4, 3, 1, -1]

#### Explanation:

#### DinnerPlates D = DinnerPlates(2); // Initialize with capacity = 2

#### D.push(1);

#### D.push(2);

#### D.push(3);

#### D.push(4);

#### D.push(5); // The stacks are now: 2 4

#### 1 3 5

#### ﹈ ﹈ ﹈

#### D.popAtStack(0); // Returns 2. The stacks are now: 4

#### 1 3 5

#### ﹈ ﹈ ﹈

#### D.push(20); // The stacks are now: 20 4

#### 1 3 5

#### ﹈ ﹈ ﹈

#### D.push(21); // The stacks are now: 20 4 21

#### 1 3 5

#### ﹈ ﹈ ﹈

#### D.popAtStack(0); // Returns 20. The stacks are now: 4 21

#### 1 3 5

#### ﹈ ﹈ ﹈

#### D.popAtStack(2); // Returns 21. The stacks are now: 4

#### 1 3 5

#### ﹈ ﹈ ﹈

#### D.pop() // Returns 5. The stacks are now: 4

#### 1 3

#### ﹈ ﹈

#### D.pop() // Returns 4. The stacks are now: 1 3

#### ﹈ ﹈

#### D.pop() // Returns 3. The stacks are now: 1

#### ﹈

#### D.pop() // Returns 1. There are no stacks.

#### D.pop() // Returns -1. There are still no stack.

#### Example 2:

**Input**:

["DinnerPlates", "push", "push", "push", "pop", "pop", "push", "push", "popAtStack", "pop", "pop", "pop"]

[[1], [10], [20], [30], [], [], [40], [50], [0], [], [], []]

**Output**:

[null, null, null, null, 30, 20, null, null, 40, 50, 10, -1]

**Explanation**:

* DinnerPlates(1) initializes with a capacity of 1.
* Then, each push operation adds a single element to the stacks.
* pop and popAtStack operations function as described, with the popAtStack popping from the specific index.

#### Constraints:

#### 1 <= capacity <= 2 \* 104

#### 1 <= val <= 2 \* 104

#### 0 <= index <= 105

#### At most 2 \* 105 calls will be made to push, pop, and popAtStack.

 **Frequently Asked By**: Amazon, Microsoft(**Year**: 2020–2023)

**Sources**:<https://leetcode.com/problems/dinner-plate-stacks/description/>

#### 19 Suppose there is a circle. There are N petrol pumps on that circle. Petrol pumps are numbered  0 to (N-1)  (both inclusive). You have two pieces of information corresponding to each of the petrol pump: (1) the amount of petrol that particular petrol pump will give, and (2) the distance from that petrol pump to the next petrol pump.

Initially, you have a tank of infinite capacity carrying no petrol. You can start the tour at any of the petrol pumps. Calculate the first point from where the truck will be able to complete the circle. Consider that the truck will stop at each of the petrol pumps. The truck will move one kilometer for each litre of the petrol.

**Input Format**

The first line will contain the value of  N.  
The next  N lines will contain a pair of integers each, i.e. the amount of petrol that petrol pump will give and the distance between that petrol pump and the next petrol pump.

**Constraints:**

1 <= N <= 105

1 <= amount of petrol,distance <= 109

**Output Format**

An integer which will be the smallest index of the petrol pump from which we can start the tour.

**Example 1:**

**Sample Input**

3

1 5

10 3

3 4

**Sample Output**

1

**Explanation**

We can start the tour from the second petrol pump.

**Asked By: Swiggy**

#### Reference :<https://www.hackerrank.com/challenges/truck-tour/problem>

#### 20 You are playing a variation of the game Zuma.

#### In this variation of Zuma, there is a single row of colored balls on a board, where each ball can be colored red 'R', yellow 'Y', blue 'B', green 'G', or white 'W'. You also have several colored balls in your hand.

#### Your goal is to clear all of the balls from the board. On each turn:

#### Pick any ball from your hand and insert it in between two balls in the row or on either end of the row.

#### If there is a group of three or more consecutive balls of the same color, remove the group of balls from the board.

#### If this removal causes more groups of three or more of the same color to form, then continue removing each group until there are none left.

#### If there are no more balls on the board, then you win the game.

#### Repeat this process until you either win or do not have any more balls in your hand.

#### Given a string board, representing the row of balls on the board, and a string hand, representing the balls in your hand, return the minimum number of balls you have to insert to clear all the balls from the board. If you cannot clear all the balls from the board using the balls in your hand, return -1.

#### Example 1:

#### Input: board = "WRRBBW", hand = "RB"

#### Output: -1

#### Explanation: It is impossible to clear all the balls. The best you can do is:

#### - Insert 'R' so the board becomes WRRRBBW. WRRRBBW -> WBBW.

#### - Insert 'B' so the board becomes WBBBW. WBBBW -> WW.

#### There are still balls remaining on the board, and you are out of balls to insert.

#### Example 2:

#### Input: board = "WWRRBBWW", hand = "WRBRW"

#### Output: 2

#### Explanation: To make the board empty:

#### - Insert 'R' so the board becomes WWRRRBBWW. WWRRRBBWW -> WWBBWW.

#### - Insert 'B' so the board becomes WWBBBWW. WWBBBWW -> WWWW -> empty.

#### 2 balls from your hand were needed to clear the board.

#### Example 3:

#### Input: board = "G", hand = "GGGGG"

#### Output: 2

#### Explanation: To make the board empty:

#### - Insert 'G' so the board becomes GG.

#### - Insert 'G' so the board becomes GGG. GGG -> empty.

#### 2 balls from your hand were needed to clear the board.

#### 

#### Constraints:

#### 1 <= board.length <= 16

#### 1 <= hand.length <= 5

#### board and hand consist of the characters 'R', 'Y', 'B', 'G', and 'W'.

#### The initial row of balls on the board will not have any groups of three or m

 **Frequently Asked By**: Google, Microsoft

####  **Year**: 2019–2023

**Sources**:<https://leetcode.com/problems/zuma-game/description/>

#### VERY HARD

#### 21. There are a number of plants in a garden. Each of the plants has been treated with some amount of pesticide. After each day, if any plant has more pesticide than the plant on its left, being weaker than the left one, it dies.

#### You are given the initial values of the pesticide in each of the plants. Determine the number of days after which no plant dies, i.e. the time after which there is no plant with more pesticide content than the plant to its left.

#### Example 1

#### p = [3,6,2,7,5]

#### // pesticide levels

#### Use a 1-indexed array. On day 1, plants  2 and  4 die leaving p’=[3,2,5] . On day 2, plant 3  in  p’ dies leaving p”=[3.2] . There is no plant with a higher concentration of pesticide than the one to its left, so plants stop dying after day 2 .

#### Function Description Complete the function poisonousPlants in the editor below.

#### poisonousPlants has the following parameter(s):

#### int p[n]: the pesticide levels in each plant

#### Returns - int: the number of days until plants no longer die from pesticide

#### Input Format

#### The first line contains an integer n , the size of the array p.

#### The next line contains n space-separated integers p[i].

#### Constraints

#### 1≤n≤105

#### 1≤p[i]≤109

#### Example 2:

#### Sample Input

#### 7

#### 6 5 8 4 7 10 9

#### Sample Output

#### 2

#### Explanation

#### Initially all plants are alive.

#### Plants = {(6,1), (5,2), (8,3), (4,4), (7,5), (10,6), (9,7)}

#### Plants[k] = (i,j) => jth plant has pesticide amount = i.

#### After the 1st day, 4 plants remain as plants 3, 5, and 6 die.

#### Plants = {(6,1), (5,2), (4,4), (9,7)}

#### After the 2nd day, 3 plants survive as plant 7 dies.

#### Plants = {(6,1), (5,2), (4,4)}

#### Plants stop dying after the 2nd day.

 **Frequently Asked By**: Facebook, Atlassian(**Year**: 2020–2023)

**Sources**:<https://www.hackerrank.com/challenges/poisonous-plants/problem?isFullScreen=true>

#### 22 You are given an integer array nums of length n and an integer array queries.

**Let gcdPairs denote an array obtained by calculating the**

**GCD**

**of all possible pairs (nums[i], nums[j]), where 0 <= i < j < n, and then sorting these values in ascending order.**

**For each query queries[i], you need to find the element at index queries[i] in gcdPairs.**

**Return an integer array answer, where answer[i] is the value at gcdPairs[queries[i]] for each query.**

**The term gcd(a, b) denotes the greatest common divisor of a and b.**

**Example 1**

**Input:** nums = [2,3,4], queries = [0,2,2]

**Output**: [1,2,2]

**Explanation:**gcdPairs = [gcd(nums[0], nums[1]), gcd(nums[0], nums[2]), gcd(nums[1], nums[2])] = [1, 2, 1].

**After sorting in ascending order, gcdPairs = [1, 1, 2].**

**So, the answer is [gcdPairs[queries[0]], gcdPairs[queries[1]], gcdPairs[queries[2]]] = [1, 2, 2].**

**Example 2:**

**Input:** nums = [4,4,2,1], queries = [5,3,1,0]

**Output:** [4,2,1,1]

**Explanation:**gcdPairs sorted in ascending order is [1, 1, 1, 2, 2, 4].

**Example 3:**

**Input:** nums = [2,2], queries = [0,0]

**Output**: [2,2]

**Explanation:gcdPairs =** [2].

**Constraints:**

* 2 <= n == nums.length <= 105
* 1 <= nums[i] <= 5 \* 104
* 1 <= queries.length <= 105
* 0 <= queries[i] < n \* (n - 1) / 2

**Asked By: Zomato**

#### Reference :<https://leetcode.com/problems/sorted-gcd-pair-queries/description/>

#### 23 Given a string containing just the characters '(' and ')', return the length of the longest valid (well-formed) parentheses substring.

#### Example 1:

#### Input: s = "(()"

#### Output: 2

#### Explanation: The longest valid parentheses substring is "()".

#### Example 2:

#### Input: s = ")()())"

#### Output: 4

#### Explanation: The longest valid parentheses substring is "()()".

#### Example 3:

#### Input: s = ""

#### Output: 0

#### Constraints:

#### 0 <= s.length <= 3 \* 104

#### s[i] is '(', or ')'.

#### Approach

#### First of all we will push into the stack the number -1.

#### now we will start the loop to get the paranthesis.

#### for (int i = 0; i < s.length(); i++) {

#### if (s.charAt(i) == '(') {

#### stack.push(i);

#### If we will get the open bracket then we will push into the stack.

#### then coming to the next:-

#### } else {

#### stack.pop();

#### if (stack.isEmpty()) {

#### stack.push(i);

#### If the stack becomes empty after popping, it means there was no unmatched '(' left. In this case, the current index i is pushed onto the stack as a new base for upcoming valid substrings.

#### else {

#### max\_len = Math.max(max\_len, i - stack.peek());

#### If the stack is not empty, it means there is an unmatched '(' available at the index of the top element. The difference i - stack.peek() calculates the length of the current valid substring, and max\_len is updated to the maximum of its current value and this new length.

#### Now lets have a example for a better understanding:-

#### For input: "(()())", the stack manipulation would look like this:

#### Initial: stack = [-1]

#### Index 0: Push index of '(', so stack = [-1, 0]

#### Index 1: Push index of '(', so stack = [-1, 0, 1]

#### Index 2: Pop (matching ')' with '('), so stack = [-1, 0] and max\_len = 2

#### Index 3: Push index of '(', so stack = [-1, 0, 3]

#### Index 4: Pop (matching ')' with '('), so stack = [-1, 0] and max\_len = 4

#### Index 5: Pop (matching ')' with '('), so stack = [-1] and max\_len = 6

#### After the loop, max\_len is 6, which is the correct length of the longest valid substring.

#### Time complexity: O(n)

#### Space complexity: O(n)

**Frequently Asked By**: Facebook, Google(**Year**: 2018–2023)

#### Sources:<https://leetcode.com/problems/longest-valid-parentheses/description/>

#### 24 You are given an integer array nums and an integer k.

**Find the longest subsequence of nums that meets the following requirements:**

* **The subsequence is strictly increasing and**
* **The difference between adjacent elements in the subsequence is at most k.**

**Return the length of the longest subsequence that meets the requirements.**

**A subsequence is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.**

**Example 1:**

**Input:** nums = [4,2,1,4,3,4,5,8,15], k = 3

**Output:** 5

**Explanation:**

The longest subsequence that meets the requirements is [1,3,4,5,8].

The subsequence has a length of 5, so we return 5.

Note that the subsequence [1,3,4,5,8,15] does not meet the requirements because 15 - 8 = 7 is larger than 3.

**Example 2:**

**Input:** nums = [7,4,5,1,8,12,4,7], k = 5

**Output:** 4

**Explanation:**

The longest subsequence that meets the requirements is [4,5,8,12].

The subsequence has a length of 4, so we return 4.

**Example 3:**

**Input:** nums = [1,5], k = 1

**Output:** 1

**Explanation:**

The longest subsequence that meets the requirements is [1].

The subsequence has a length of 1, so we return 1.

**Constraints:**

* 1 <= nums.length <= 105
* 1 <= nums[i], k <= 105

**Intuition**

Implement a naive DP, namely if we have state dp[i][value] be the best we can do with the first i elements and such that the last element is value. Then it is not hard to see that

dp[i][nums[i]]=1+max

v∈[nums[i]−k,nums[i])dp[i][v].

But this naive approach is only O(n2 ) so is not fast enough. We want at least O(nlogn). But thankfully, we can note that:

* In the transition from dp[i-1] to dp[i] the only value that can change is dp[i][nums[i]]. So we dont need to maintain a whole dimension of DP.
* We need to process find max of range (RMQ) quickly -- for this we can use Segment Tree.

In fact, we do not need a dp array at all, we need only maintain a segment tree and update it for each iteration over nums.

**Approach**

Maintain a segment tree of size such that it can handle queries up to 1e5+1. Here we use a segment tree that handles queries for segments [l,r). Then iterate over nums, updating the segment tree as per the following sudo code:

for num in nums:

best = 1 + RMQ(num - k, num)

Update(num, best)

return RMQ(0, 1e5+1)

We can look at the Naive DP to see why this works. The segment tree is a so-called "bottom up" segment tree where the nodes are stored in heap-like format where child of tree[i] is tree[2\*i], tree[2\*i+1]. This form of segment tree is known to be much faster than the normal recursive implementation (sometimes even faster than Fenwick Tree).

**Time complexity:**

O(nlogN), where n is the lengths of nums and N is max value of nums (but in implementation, we just use 1e5+1). Both update and query operations on a segment tree are O(logN).

**Space complexity:**

O(N), where N is max of nums.

**Asked By : Google,Microsft,Flikart**

**Reference :** [**https://leetcode.com/problems/longest-increasing-subsequence-ii/description/**](https://leetcode.com/problems/longest-increasing-subsequence-ii/description/)

#### 25 Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

#### https://assets.leetcode.com/uploads/2018/10/22/rainwatertrap.png

#### Example 1:

#### Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]

#### Output: 6

#### Explanation: The above elevation map (black section) is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.

#### Example 2:

#### Input: height = [4,2,0,3,2,5]

#### Output: 9

#### Constraints:

#### n == height.length

#### 1 <= n <= 2 \* 104

#### 0 <= height[i] <= 105

 **Frequently Asked By**: Amazon, Microsoft( **Year**: 2017–2023)

#### Sources:<https://leetcode.com/problems/trapping-rain-water/description/>