Lesson:

Problems based on Stacks

PW SKILLS

Pre-Requisites

- Basics of stacks

Today's Checklist:

- Problems related to Stacks

Topic 1: Problems related to stacks

Balanced bracket sequence

Q1. Check if a given bracket sequence consisting of '(' and ')' is balanced or not. A balanced bracket sequence is a string consisting of only brackets, such that this sequence, when inserted with certain numbers and mathematical operations, gives a valid mathematical expression.

Input
)(
Output
false

Input
)(())
Output
false

Code

Explanation

1. Traverse through the input string.
2. At each iteration-
   a. If the current character is an opening bracket, push it into the stack.
   b. If the current character is a closing bracket, see if there exist a opening bracket in the stack that corresponds to this bracket-
      i. If the stack is empty, return false.
      ii. Else we have found a matching pair so remove the opening bracket from the stack.
3. If at the end the stack has unresolved brackets, return false.
4. Else return true.

Next greater element

Q2. Given an arraylist, print the Next Greater Element (NGE) for every element.
The Next Greater Element for an element x is the first greater element on the right side of x in the vector.
Elements for which no greater element exists, consider the next greater element as -1.
The first line of input contains the size of the vector.
The second line of input contains the elements of the vector.

Input
4
1 3 2 4

Output
3 4 4 -1

Code

Approach

1. Traverse the elements of the vector from right to left.
2. While the element on the top of the stack is less than the current element of the vector, pop the top of the stack.
3. If the stack is not empty, the top of the stack is the next greater element of the current element.
4. Push the current element into the stack.

Stock span problem

Q3. Given a series of N daily price quotes for a stock, we need to calculate the span of the stock's price for all N days. The span of the stock's price in one day is the maximum number of consecutive days (starting from that day and going backward) for which the stock price was less than or equal to the price of that day.

The first line of input contains the value of N.
The second line of input contains the stock price for each day.

Input
7
100 80 60 70 60 75 85
Output
1 1 1 2 1 4 6

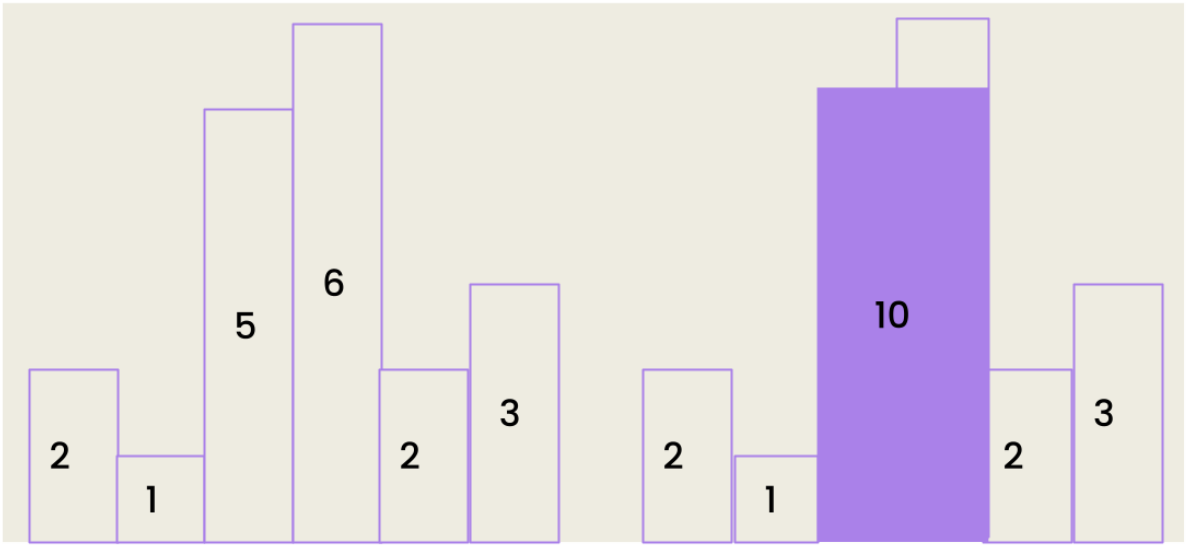
Code

Approach
The answer for any element is the distance between the current element and the next greater element on the left.

1. Traverse the list from left to right.
2. While the element at the index on the top of the stack is less than or equal to the current element of the vector, pop the top of the stack.
3. If there is any element in the stack, push the difference between the current index and the index on the top of the stack into the answer vector.
4. Else, push current index + 1 into the stack (-(-1) is +1).

Largest rectangle in histogram

Q4. Given an array of integer heights representing the histogram's bar height where the width of each bar is 1, return the area of the largest rectangle in the histogram.
The first line of input contains the number of elements in the array.
The second line of input contains the elements of the array.



Input
6
2 1 5 6 2 3
Output
10

Approach - 1

Steps

1. For each index, we can find the next smaller element on the left and the next smaller element on the right. The distance between the two, both exclusive) will make up the width of the rectangle that can be formed with the current height.Let's take the above example. Let's say we are at index 2. For this index, the next greater element on the left is 1 and the next greater element on the right is 2. As a result the max width for the rectangle with the height 5 will be 2 (the distance between the block of height 1 and height 2).
2. First find the next smaller element for each index on the left hand side and store it in a vector.
3. Then find the next greater element for each index on the right hand side and using the left hand next smaller element, find the maximum area for the current index.
4. The answer will be the maximum of the areas for each index.

Code

Approach - 2

Steps

1. When a bar is popped, we calculate the area with the popped bar at index idx as the shortest bar. Now we know the rectangle height is heights[idx], we just need rectangle width to calculate the area.
2. How to determine rectangle width? The maximum width we can have here would be made of all connecting bars with height greater than or equal to heights[idx]. heights[s.peek() + 1] >= heights[idx] because the index on top of the stack right now s.peek() is the first index left of idx with height smaller than idx's height (if s.peek() was greater then it should have already been popped out of the stack). heights[i - 1] >= heights[idx] because index i is the first index right of idx with height smaller than idx's height (if i was greater then idx would have remained on the stack). Now we multiply height heights[idx] by width i - 1 - s.peek() to get the area.

Code