← Back to Home

Two Pointers
Sliding Window
Intervals
Stack
- Overview
- Valid Parentheses
- Decode String
- Longest Valid Parentheses

Monotonic Stack
- Overview
- Daily Temperatures
- Largest Rectangle in Histog...

Linked List
Binary Search
Heap
Depth-First Search
Breadth-First Search
Backtracking
Graphs
Dynamic Programming
Greedy Algorithms

← Overview

Largest Rectangle i... →

Daily Temperatures

DESCRIPTION (credit Leetcode.com)

Given an integer array `temps` representing daily temperatures, write a function to calculate the number of days one has to wait for a warmer temperature after each given day. The function should return an array answer where `answer[i]` represents the wait time for a warmer day after the `ith` day. If no warmer day is expected in the future, set `answer[i]` to 0.

EXAMPLES

Inputs:

```
temps = [65, 70, 68, 60, 55, 75, 80, 74]
```

Output:

```
[1,4,3,2,1,1,0,0]
```

```python
class Solution:
    def dailyTemperatures(self, temps: list[int]):
        # Your code goes here
        pass
```

Results AI Feedback View Answer Run

Run your code to see results here

Explanation

This question uses a monotonically decreasing stack to find the next greatest temperature for each day in $O(n)$ time, compared to the $O(n^2)$ time of the brute-force approach.

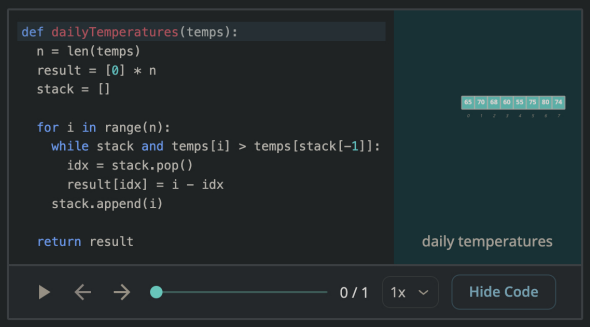
A monotonically decreasing stack is one where all elements decrease in value (from the bottom of the stack to the top). When pushing an item on a monotonically decreasing stack, it must be smaller than all the other elements that are currently on the stack.

On This Page

Daily Temperatures
Explanation
Solution
- Complexity Analysis

A montonically decreasing stack

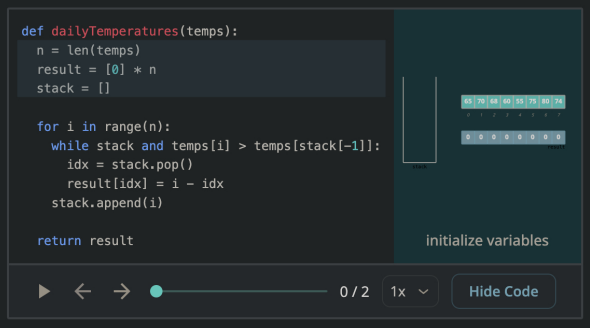
First, we initialize our stack and our results array. Our stack holds indices of the temperatures in the input array that are waiting for a higher temperature, and our results array holds the number of days we have to wait after the `ith` day to get a warmer temperature.



Next, we iterate over each index in the array. For each index, we get the current temperature of that index, and compare it to the temperature of the top index in the stack.

Pushing To The Stack

If the current temperature is less than the top temperature in the stack (of if the stack is empty), we push the current index onto the stack to indicate that we are waiting to find a greater temperature for that index.



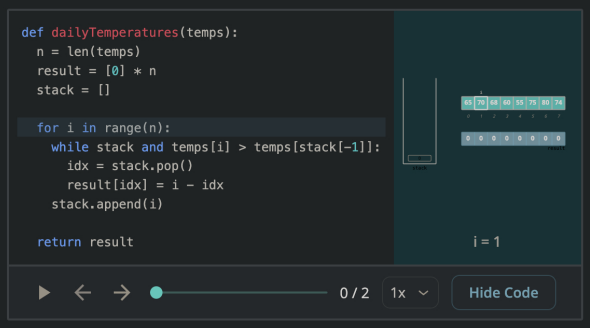
Pushing index 0 to the stack

Popping From The Stack

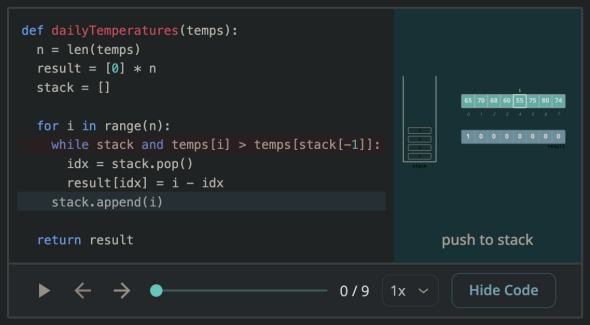
When the current temperature is greater than the top temperature in the stack, we have found the next highest temperature for not only the top index in the stack, but potentially other indices in the stack as well, which we can efficiently process due to the monotonically decreasing stack.

We first pop the top index from the stack and calculate the number of days we had to wait for that popped index to find a warmer temperature (current index minus the popped index), and store that number in the results array at the index of the popped element. To account for the fact that the current temperature might be the next greatest temperature for multiple indicies, we repeat this process in a while loop until the current temperature is less than the top temperature in the stack, or until the stack is empty.

After that is done, we push the current index onto the stack to indicate that we have not yet found the next greatest temperature for the current index.



Popping index 0 from the stack

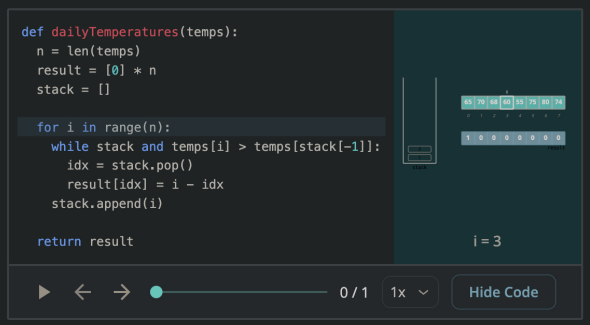


Popping multiple indexes from the stack. 75 is a higher temperature for 55, 60, 68 and 70.

This continues until we have iterated over the entire input array. At the end, we return the results array.

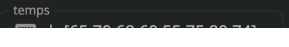
Efficency of the Monotonically Decreasing Stack

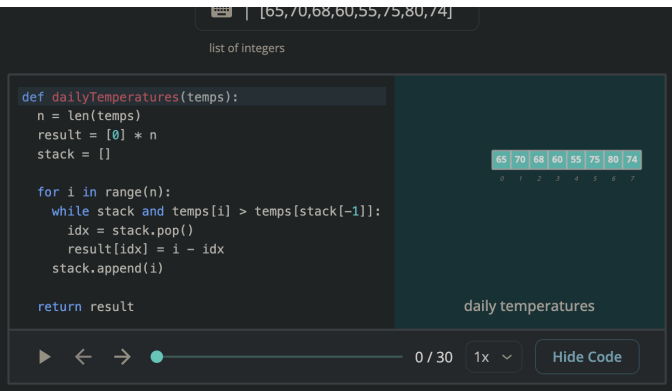
This is more efficient than the brute-force approach because it reduces the number of comparisons we have to make. For example, consider the state of the stack when we are at the 3rd index in the input array.



Because the stack is monotonically decreasing, we only have to compare the temperature at `i` (60) to the temperature of the index at the top of the stack (68), to know that it cannot be a higher temperature for the other remaining items on the stack (70), which avoids a comparision between 60 and 70. But in the brute-force approach, 60 and 70 are compared when finding the next greatest temperature after 70.

Solution





Complexity Analysis

Time Complexity: `O(n)` . Each item is pushed and popped from the stack at most once.

Space Complexity: `O(n)` where `n` is the length of the input array for the output `results` array.

Next: Largest Rectangle in Histogram →

Add a comment...

Anonymous

Post As Tariq Williams

OriginalAquamarineErmine112
Created at: 9/6/2024, 11:21 PM

The example output for the question should be [1, 4, 3, 2, 1, 1, 0, 0] not [1, 4, 1, 2, 1, 1, 0, 0]

Reply

Jimmy Zhang
Created at: 9/7/2024, 10:14 AM

Thanks for catching that. will be fixed shortly

Reply