901. Online Stock Span

Problem Statement

Design an algorithm that collects daily price quotes for some stock and returns **the span** of that stock's price for the current day.

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.

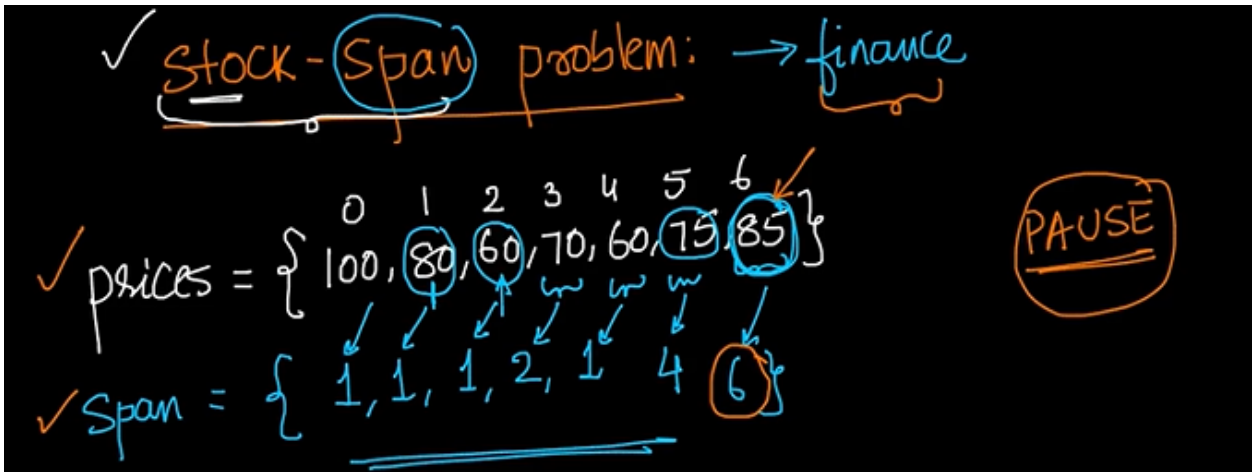
- For example, if the prices of the stock in the last four days is `[7,2,1,2]` and the price of the stock today is `2`, then the span of today is `4` because starting from today, the price of the stock was less than or equal `2` for `4` consecutive days.
- Also, if the prices of the stock in the last four days is `[7,34,1,2]` and the price of the stock today is `8`, then the span of today is `3` because starting from today, the price of the stock was less than or equal `8` for `3` consecutive days.

Implement the `StockSpanner` class:

- `StockSpanner()` Initializes the object of the class.
- `int next(int price)` Returns the **span** of the stock's price given that today's price is `price`.

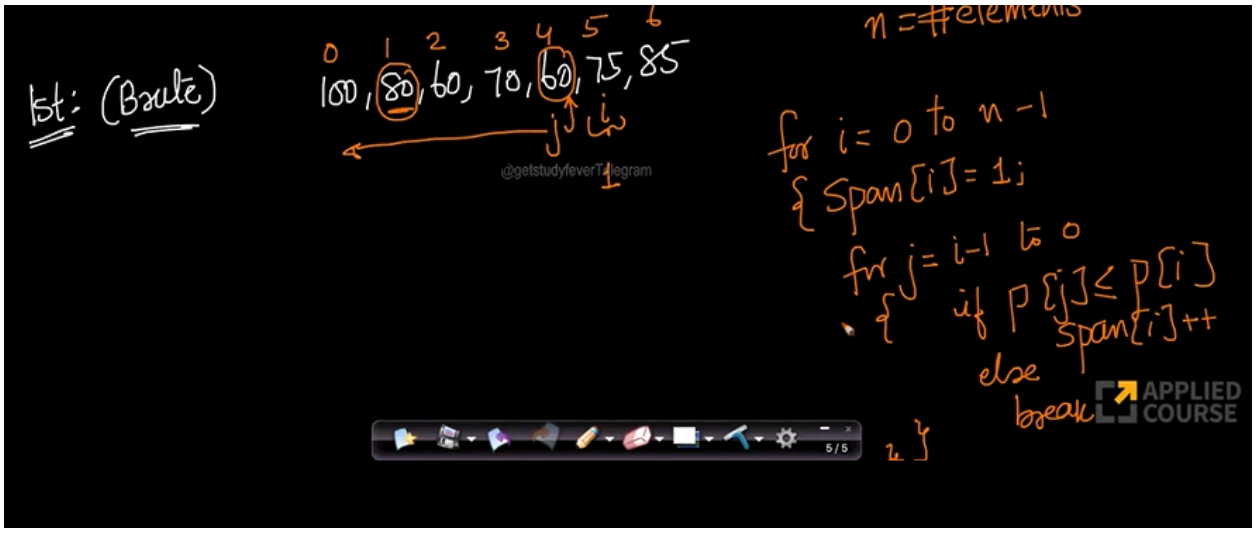
Example 1:

Input
```
["StockSpanner", "next", "next", "next", "next", "next", "next", "next"]
[[], [100], [80], [60], [70], [60], [75], [85]]
```
Output
```
[null, 1, 1, 1, 2, 1, 4, 6]
```



1st approach

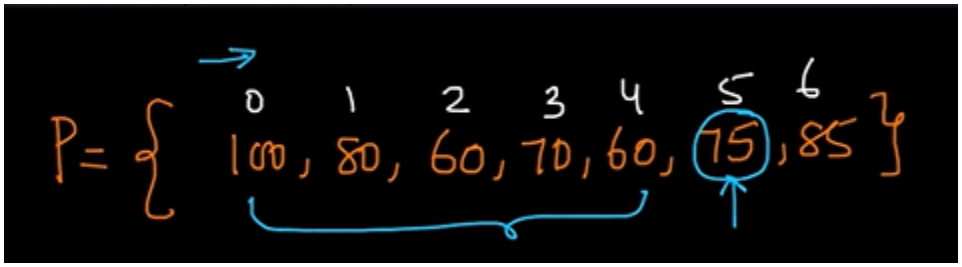
1. First approach is just to traverse from back..and in another loop traversing the remaining array



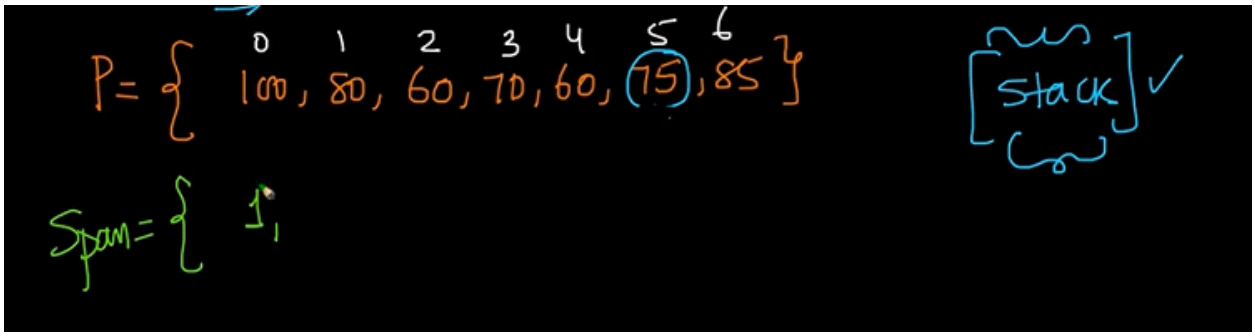
2. This approach takes O(n^2) and O(1)...

Optimized Approach

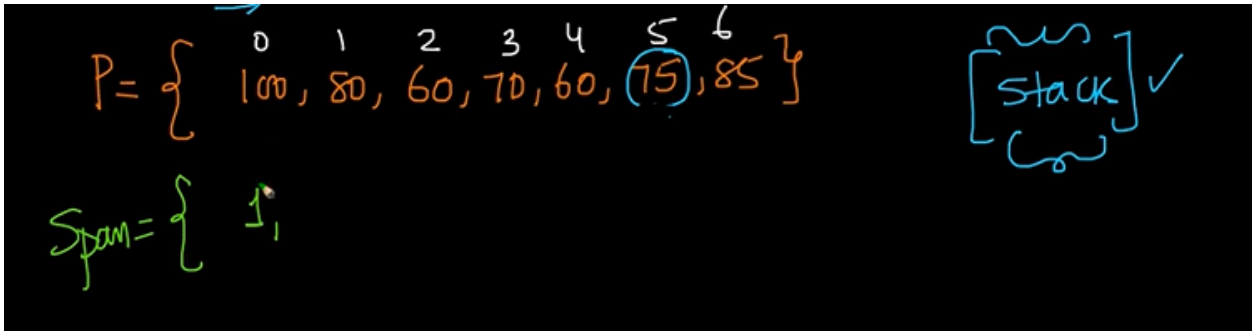
1. Here we'll be using stack



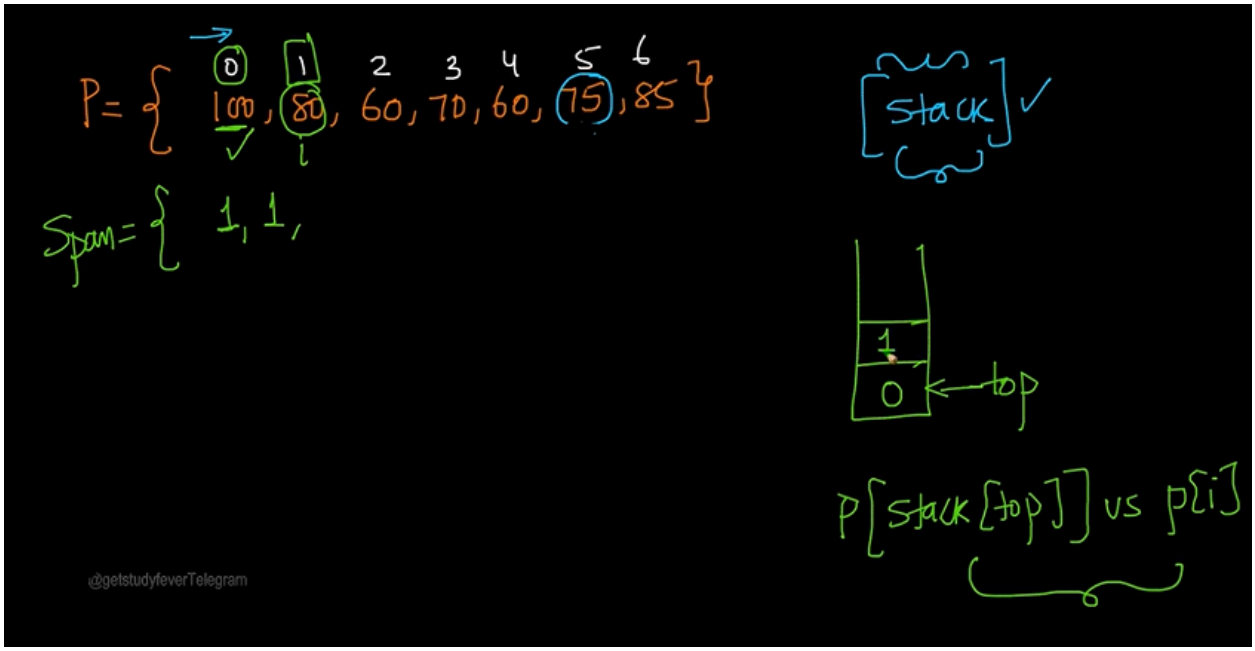
2. So now to find the span of 75…we need to know its previous values
3. Here we'll start from left to right
4. So here for the first element the span is 1 by default



5. Now in stack we'll store the information of elements(we'll store index of elements) whose span is already calculated..here by default index 0(100) span 1…so we push index 0 to the stack

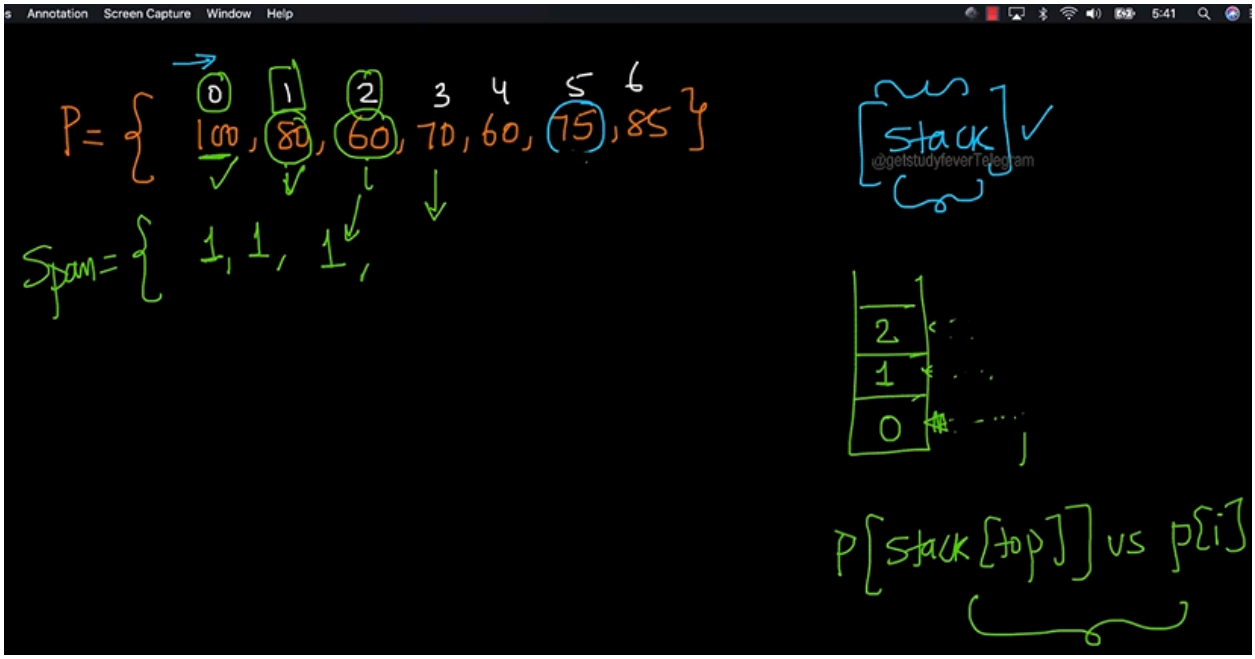


6. Next We move to index 1..and in stack our top element is P[stack[top]] now we compare …top element with P[i]....if top element is greater than current element (p[i])...then the span of current element is (current_index-top) and we push our index to the stack and

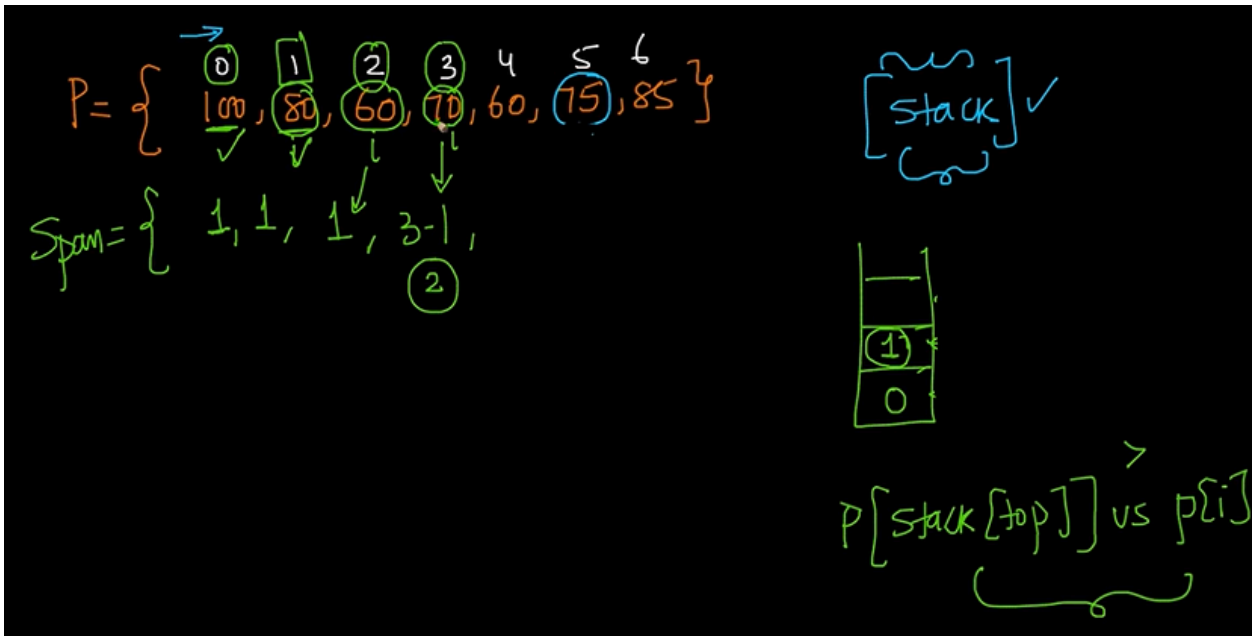


7. Next we move to index 2 and the current value is 60…compare it with stack's top (80) as stack's top is greater than current element then span of current element is current_index

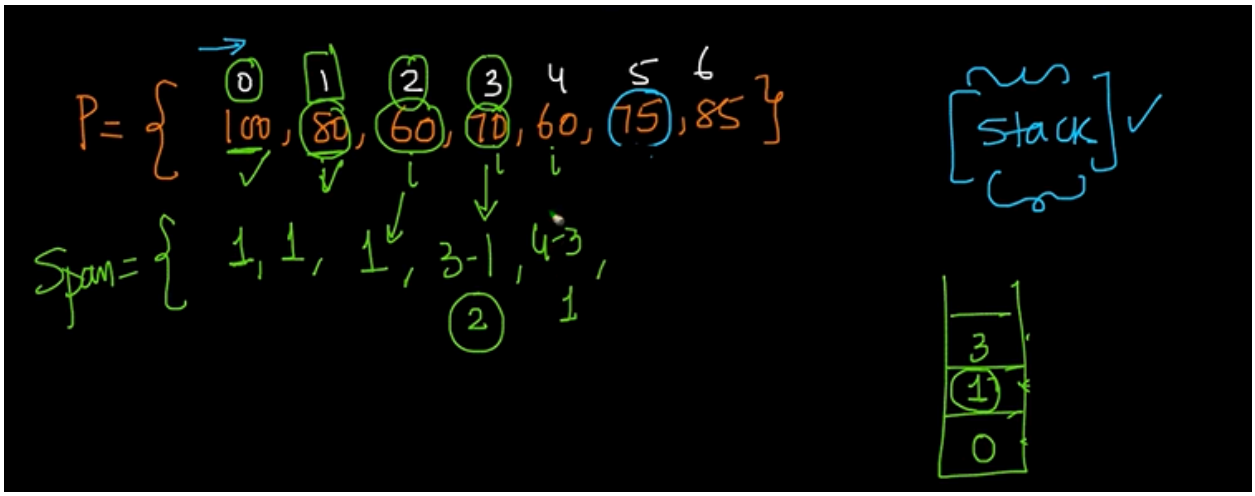
- top

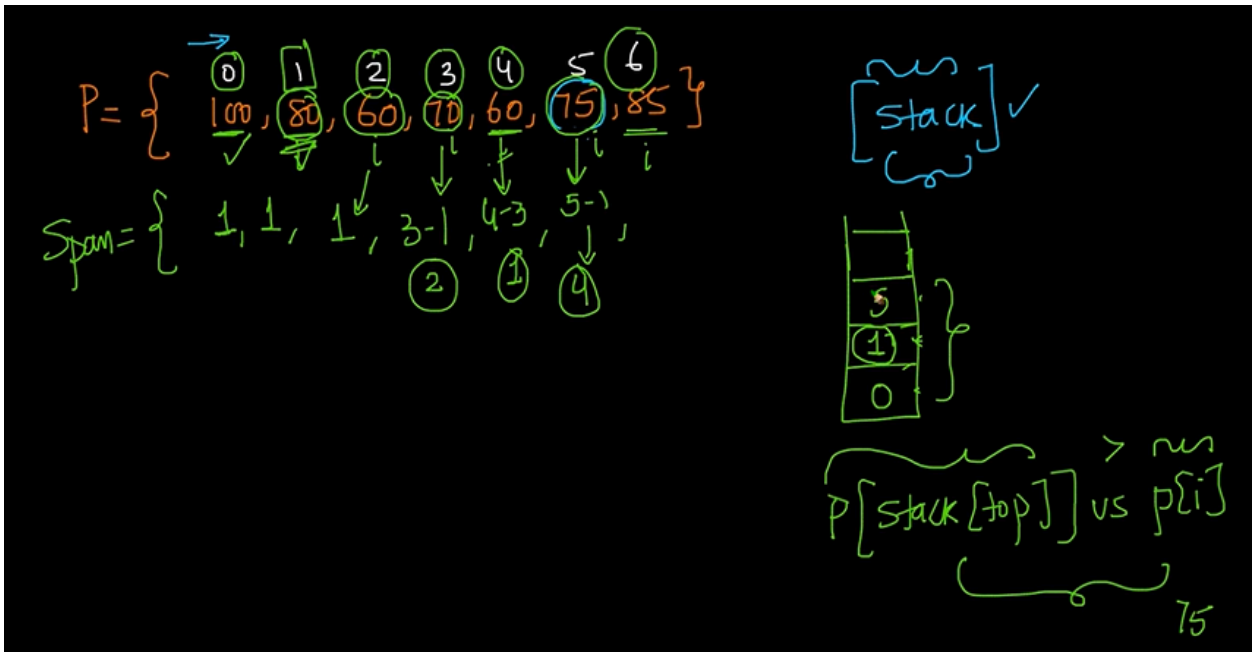


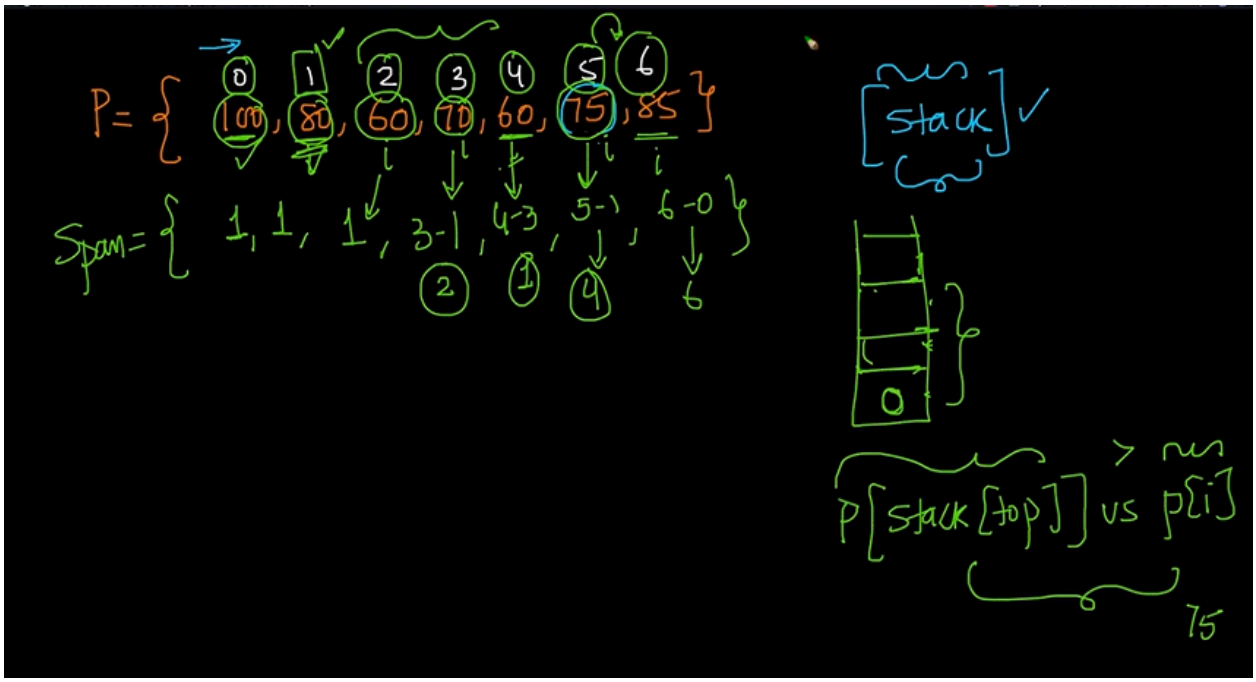
8. Now we'll move to next index which is index 3 and current value is 70…P[i]>Stack[top]...so we pop() the top element which is 2…and we compare it with the current top (which is 1) and it fails…so span of 70 is 3-1 = 2



9.


P = { 100, 80, 60, 70, 60, 75, 85 }
0 1 2 3 4 5 6
Span = { 1, 1, 1, 3-1, 4-3, 2 1
stack
3
1
0


P = { 100, 80, 60, 70, 60, 75, 85 }
0 1 2 3 4 5 6
Span = { 1, 1, 1, 3-1, 4-3, 5-1, 2 1 4
stack
5
1
0
P[stack[top]] vs p[i]
75



10. Similarly we'll traverse all the elements in the array and find span