CS202 - Algorithm Analysis

Data Structure & Algorithm 1

Aravind Mohan

Allegheny College

January 30, 2023

A Follow-up from last week

5	5	5	9	5	5	5	5

```
def detect(coins):
  pos = 0
  # add your logic here
  for i in range(0, len(coins)-1):
    if (coins[i] != coins[i+1]):
      if (coins[i] < coins[i+1]):
        pos = i
      else:
        pos = i+1
  return pos
```

Python

Discussion Based On ...

Sedgewick 1.3, Stacks

Stack

A plate dispenser is like a **Stack**.

What is a Stack ADT?

- A stack is a container of objects that are inserted and removed according to the last-in-first-out (**LIFO**) principle.
- Objects can be inserted at any time, but only the last (the most-recently inserted) object can be removed.
- Inserting an item is known as ”pushing” onto the stack. ”Popping” off the stack is synonymous with removing an item.

Stack ADT Operations

- A stack is an Abstract Data Type that supports four main methods:
  1. **new():**ADT - Creates a new stack.
  2. **push(S:ADT, o:element):**ADT - Inserts object o onto top of stack S.
  3. **pop(S:ADT):**ADT - Removes the top object of stack S; if the stack is empty an error occurs.
  4. **top(S:ADT):**element - Returns the top object of the stack, without removing it; if the stack is empty an error occurs.

Stack ADT Supporting Operations

- **size(S:ADT):**integer - Returns the number of objects in stack S.
- **isEmpty(S:ADT):**boolean - Indicates if stack S is empty.

Axioms on Stacks

An axiom, is a statement that is taken to be true, to serve as a premise or starting point for further reasoning and arguments.

The following axioms dictates the scope of the operations in the stack.

- **Pop(Push(S,v)) = S**
- **Top(Push(S,v)) = v**

Algorithmic Problem

Ready for an Algorithmic Problem?

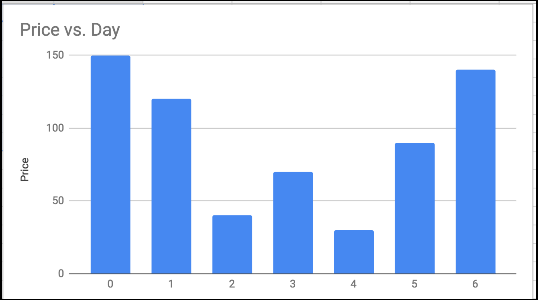
Solve it correctly first and then Solve it **fast**.

Time Series Problem

The span S_i of a stock's price on a certain day i is the maximum number of consecutive days (up to the current day) the price of the stock has been less than or equal to its price on day i.

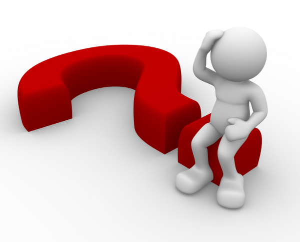
Let us suppose we are given with the stock prices for different days. Identify the span S for those days.

Time Series Problem



Span[0,1,2,3,4,5,6] = $\{1,1,1,2,1,4,6\}$

So how to write an algorithm to solve this problem?

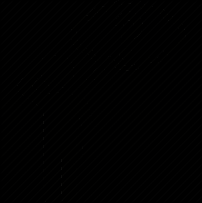


Please dont see next slide. **Think** yourself first!

Time Series Algorithm - A first attempt

```
Algorithm ComputeSpan(P):
Input: an n-element array P of numbers such that
       P[i] is the price of the stock on a day i.
Output: an n-element array S of numbers such that
       S[i] is the span of the stock on a day i.

for i = 0 to n do
  h = 0, done = false
  repeat
    if P[i-h] <= P[i] then
      h = h+1
    else
      done
  until (h = i) or done
  S[i] = h
return S
```



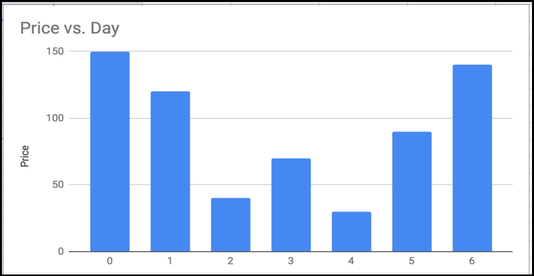
- What is the worst-case asymptotic running time of this algorithm?
- How to transform this algorithm into its code/program equivalent?

Next: Is it possible to solve this efficiently using a Stack ADT? and in a **LINEAR time**

Time Series Problem

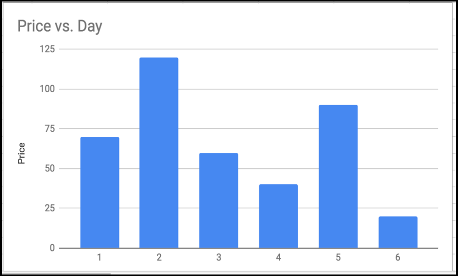
Approach: S[i] can be easily computed if we know the closest day preceding i on which the price is greater than the price on i. If such a day exist let's call the day as h_i, otherwise we conventionally define h_i = -1.

Time Series Problem



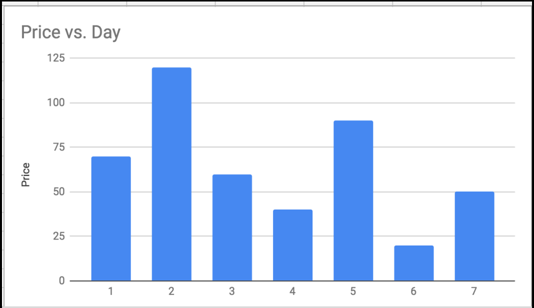
- h(3) = 1; h(4) = 3; h(5) = 1;
- **What is h(6)?**
- What is the formula to calculate span S[i]?
- $S[i] = i - h(i)$

Time Series Problem



- What are possible values of h(7)? Can it be 1, 3, or 4?
- No. h(7) can be 2 or 5 or 6
- We store indices 2, 5, 6 in the stack
- To determine the price of h(7) we compare the price on day 7 with the day 6, day 5 and day 2 in that order.

Time Series Problem



- The first price larger than the day 7 gives h(7). The stack should be updated to reflect the price of day 7. It should now contain
  2, 5 ,7.

The Efficient Algorithm

Algorithm - ComputeSpan(P)

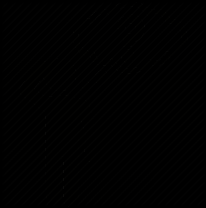
Input: an n-element array P of numbers such that P[i] is the price of the stock on a day i.

Output: an n-element array S of numbers such that S[i] is the span of the stock on a day i.

The Efficient Algorithm

```
Initialize an empty stack D
Initialize an array S
for i = 0 to n − 1 do
    h ← 0
    done ← false
    while not (D is empty() or done) do
        if P[i] >= P[D.top()] then
            D.pop()
        else
            done ← true
        end if
    end while
    if D is empty() then
        h ← −1
    else
        h ← D.top()
    end if
    S[i] ← i − h
    D.push(i)
end for
return S
```

Thinking Exercise



- What is the worst-case asymptotic running time of this algorithm?
- How to transform this algorithm into its code/program equivalent?

Inclusion of a Stack Data Structure has a positive effect on the performance of an algorithm.

Reading Assignment

Sedgewick 1.3 Stack

Questions?

Please ask if there are any Questions!