Computing Spans (not in book)

- We show how to use a stack as an auxiliary data structure in an algorithm.
- Given an array $X$, the span $S[i]$ of $X[i]$ is the maximum number of consecutive elements $X[j]$ immediately preceding $X[i]$ and such that $X[j] \leq X[i]$.
- Spans have applications to financial analysis.
  - E.g., stock at 52-week high.

Computing Spans with a Stack

- We keep in a stack the indices of the elements visible when "looking back".
- We scan the array from left to right.
  - Let $i$ be the current index.
  - We pop indices from the stack until we find index $j$ such that $X[i] < X[j]$.
  - We set $S[i] = i - j$.
  - We push $x$ onto the stack.

Quadratic Algorithm

**Algorithm spans1($X, n$)**

```
Input array $X$ of $n$ integers
Output array $S$ of spans of $X$
$S \leftarrow$ new array of $n$ integers
for $i \leftarrow 0$ to $n - 1$ do
  $s \leftarrow 1$
  while $s \leq i$ and $X[i] - s \leq X[i]$
    $s \leftarrow s + 1$
  $S[i] \leftarrow s$
return $S$
```

Algorithm spans1 runs in $O(n^2)$ time.

Linear Algorithm

**Algorithm spans2($X, n$)**

```
S \leftarrow$ new array of $n$ integers
A \leftarrow$ new empty stack
for $i \leftarrow 0$ to $n - 1$ do
  while ($A$ is empty) and $X[top()] \leq X[i]$ do
    $A.pop()$
    if $A$ is empty
      $S[i] \leftarrow i + 1$
    else
      $S[i] \leftarrow i - \text{top}()$
    $A.push()$
$A.push()$
return $S$
```

Algorithm spans2 runs in $O(n)$ time.