Topcoder SRM 647, D1, 250-Pointer
"BuildingTowersEasy"

James S. Plank
EECS Department
University of Tennessee

CS494 Class
January 21, 2016
You are building a numbered sequence of towers.

The heights of consecutive towers may differ by at most one.

Tower one's height is zero.
The problem

- You are given some constraints:
  - The number of towers.
  - The maximum heights of some subset of the towers.

- Report the height of the tallest tower where all of the towers meet the constraints.

**Example 0:**
- 10 towers.
- #3 and #8 have a max height of one.
- The answer is 3.
Prototype

- **Class name:** BuildingTowersEasy
- **Method:** maxHeight()
- **Parameters:**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>int</td>
<td>Number of towers</td>
</tr>
<tr>
<td>$x$</td>
<td>vector &lt;int&gt;</td>
<td>Id's of towers with maximum heights</td>
</tr>
<tr>
<td>$y$</td>
<td>vector &lt;int&gt;</td>
<td>The maximum heights</td>
</tr>
</tbody>
</table>

- **Return Value:** int
Constraints

- $N \leq 1,000,000$.
- $x$.size() $\leq 50$ (and $\leq N$).
- $x$ is increasing.
- $t[i] \leq 1,000,000$
- There is a legal answer.
Giving the problem some thought.

- Because `x.size() <= 50`, an \(O(n^3)\) algorithm will work fine.

- **Question #1**: when will tower \(x[i]\) have a lower height than \(t[i]\)?

- **Answer**: When it is constrained by some other tower \(x[j]\) with a height of \(t[j]\).
Giving the problem some thought.

- Because `x.size() <= 50`, an $O(n^3)$ algorithm will work fine.

- **Question #1**: when will tower $x[i]$ have a lower height than $t[i]$?

- **Answer**: When it is constrained by some other tower $x[j]$ with a height of $t[j]$.

Suppose:

$x = \{3, 6\}$
$t = \{1, 10\}$

Tower 6 is constrained by Tower 3.
Giving the problem some thought.

- Because `x.size() <= 50`, an $O(n^3)$ algorithm will work fine.

- **Question #1**: when will tower $x[i]$ have a lower height than $t[i]$?

- **Answer**: When it is constrained by some other tower $x[j]$ with a height of $t[j]$.

Suppose:

- $x = \{3, 4, 6\}$
- $t = \{1, 10, 10\}$

Towers 4 and 6 are both constrained by Tower 3.
Step #1.

- Find out the actual height of each $x[i]$.
- Do that by scanning each other $t[i]$.
- (BTW, sentinelize towers 1 and $N$ if you need to).
- This is $O(n^2)$, where $n = x.size()$.

Suppose:

$N = 7$

$x = \{1, 3, 4, 6, 7\}$

$T = \{0, 1, 10, 10, 100000\}$

Actual $= \{0, 1, 2, 4, 5\}$
Step #2

- Between each pair of adjacent values of $x$, figure out the maximum tower between the two towers.
- This is going to be a function of $|x[i] - x[i-1]|$ and $|t[i] - t[i-1]|$. 
Running Time:

- Step 1 is $O(n^2)$, where $n = x.size()$.
- Step 2 is $O(n)$.
- Therefore, the total is $O(n^2)$. 
Can you make it faster?

- Yes – sort towers by \( t[i] \) and insert them in increasing order into a map keyed on tower number.
- When you insert a tower into the map, the only towers that can constrain it are the ones to its left and its right in the map.
- Makes the running time \( O(n \log(n)) \).

Suppose:

\[ N = 11 \]
\[ x = \{2, 5, 7, 8, 10\} \]
\[ T = \{1, 10, 9, 11, 1\} \]
I should have an experiment here.
How did the Topcoders Do?

- A pretty average problem:
  - 509 Topcoders opened the problem.
  - 434 (85%) submitted a solution.
  - 370 (85%) of the submissions were correct.
  - (That's a 73% success rate)
  - Best time was 2:40.
  - Average correct time was 27:34.
  - (I did this one live: 22 minutes for 169.26 pts).