Binary Search

k = 48?

low | mid | high

1 2 5 10 15 20 22 30 35 40 43 48 51

k = 48?
### Binary Search

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>22</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>43</th>
<th>48</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>mid</td>
<td>high</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

k = 48?

### Binary Search

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>22</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>43</th>
<th>48</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>mid</td>
<td>high</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

k = 48?

---

https://research.googleblog.com/2006/06/extra-extra-read-all-about-it-nearly.html

“The version of binary search that I wrote for the JDK (java.util.Arrays) contained the same bug. It was reported to Sun recently when it broke someone’s program, after lying in wait for nine years or so.”

---

Extra, Extra - Read All About It: Nearly All Binary Searches and Mergesorts are Broken

Friday, June 02, 2006

Posted by Joshua Bloch, Software Engineer

```java
int mid = (low + high) / 2;
```
Recurrence relations

- By itself, a recurrence does not describe the running time of an algorithm
  - need a **closed-form solution** (non-recursive description)
  - exact closed-form solution may not exist, or may be too difficult to find

- For most recurrences, an asymptotic solution of the form $\Theta()$ is acceptable
  - ... in the context of analysis of algorithms

How to solve recurrences?

- By **unrolling** (expanding) the recurrence
  - a.k.a. **iteration method** or repeated substitution

- By **guessing** the answer and proving it correct by **induction**

- By using a **Recursion Tree**

- By applying the **Master Theorem**

Analysis of Binary Search

- **Base Case**
  \[ T(1) = c_0 \]

- **Recursive Case**
  \[ T(n) = T(n/2) + c_1 \]

“A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.” (CLRS)
Unrolling a Recurrence

- Keep unrolling the recurrence until you identify a general case
  - then use the base case

- Not trivial in all cases but it is helpful to build an intuition
  - may need induction to prove correctness

Applying the base case

We already know $T(1)$ is equal to a constant $c_0$:

$$T(n) = T(n/2) + c_1$$

$$= T(n/2^k) + kc_1$$

$$= c_0 + c_1 \log n$$

$$= O(\log n)$$

Recursion Tree (binary search)

$$T(1) = c_0$$

$$T(n) = T(n/2) + c_1$$

$$= T(n/2^k) + kc_1$$
Recursion Tree (binary search)

\[ T(1) = c_0 \]
\[ T(n) = T(n/2) + c_1 \]

\[ = c_0 + c_1 \log n \]
\[ = O(\log n) \]

---

Example

\[ T(1) = a \]
\[ T(n) = 2T(n/2) + n \]

Example

\[ T(1) = 1 \]
\[ T(n) = T(n-1) + c \]

Example

\[ T(0) = 0 \]
\[ T(n) = 2T(n-1) + 1 \]
Example

```c
int power(int b, int n) {
    // base case
    if (n == 0) {
        return 1;
    }
    // recursive call
    return b * power(b, n-1);
}
```

Unimodal arrays (find max)

• An array is **strongly unimodal** if it can be split into an increasing part followed by a decreasing part

```
1 2 5 8 15 20 22 20 15 12 10 8 5
```

• An array is **weakly unimodal** if it can be split into a nondecreasing part followed by a nonincreasing part

```
1 2 5 5 15 20 22 22 35 38 13 8 5
```

Unimodal arrays

• An array is **strongly unimodal** if it can be split into an increasing part followed by a decreasing part

```
1 2 5 8 15 20 22 20 15 12 10 8 5
```

• An array is **weakly unimodal** if it can be split into a nondecreasing part followed by a nonincreasing part

```
1 2 5 5 15 20 22 22 35 38 13 8 5
```

Find the **max** (strongly unimodal)

```
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
1 2 5 8 15 20 22 20 15 12 10 8 5
```
Find the **max** (strongly unimodal)

- Algorithm?

- Running Time?

---

Find the **max** (weakly unimodal)

- Algorithm?

- Running Time?

- Recursion Tree?

Two recursive calls