Clever Uses of Binary Search

Section 3.3

Dr. Mayfield and Dr. Lam

Department of Computer Science
James Madison University

Oct 23, 2015

Guessing games

- I'm thinking of a number between 1 and 100

- I'm thinking of a last name of a JMU student

STL implementation

```
#include <algorithm>
typedef T int; // or string, whatever

    T value;
    bool found;
    vector<T> v;
    ...
    sort(v.begin(), v.end()); // don't forget to sort!
    found = binary_search(v.begin(), v.end(), value);
```

http://www.cplusplus.com/reference/algorithm/binary_search/

Java implementation

```
import java.util.*;


    T value;
    int index;
    List<T> v;
    ...
    Collections.sort(v); // don't forget to sort!
    index = Collections.binarySearch(v, value);
    // returns a negative value if not found
```

http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html

Iterative solutions

Basic idea:

```
hi = initial_high_guess();
lo = initial_low_guess();
while (hi > lo)
{
    mid = (hi - lo) / 2 + lo;
    if (is_less_than(mid)) {
        hi = mid;          // answer is in [lo,mid)
    } else {
        lo = mid;          // answer is in [mid,hi]
    }
}
```

Trick:

```
mid = (hi - lo) / 2 + lo;    // intuitive
mid = (hi + lo) / 2;         // same but quicker
                             // and less error-prone
```

Iterative solution #1

```
// adjustable threshold
#define EPSILON 1e-9

    while (hi - lo > EPSILON)
    {
        double mid = (lo + hi) / 2.0;
        if (/* depends on the problem */)
            hi = mid; // guess lower
        else
            lo = mid; // guess higher
    }
    return hi;
```

Be careful with floating point precision

- Correct up to n decimal places
- Remember that `printf` rounds

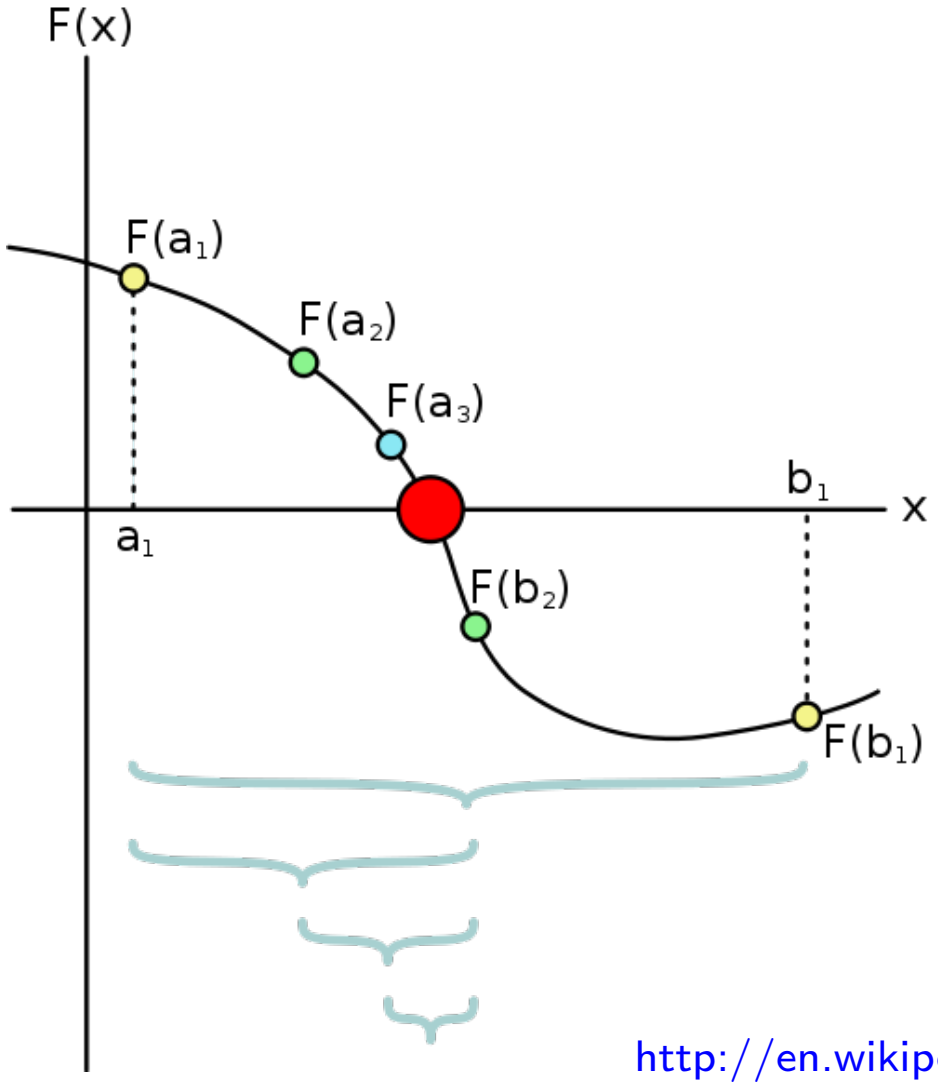
Iterative solution #2

```
for (i = 0; i < 50; i++)
{
    double mid = (lo + hi) / 2.0;
    if (/* depends on the problem */)
        hi = mid; // guess lower
    else
        lo = mid; // guess higher
}
return hi;
```

The book points out that $log_2((10000 - 0)/10^{-9}) \approx 43$

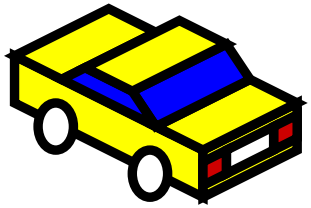
- Constant number of iterations avoids precision errors
- Avoids the case when #1 results in an infinite loop
- Plus you don't need the check overhead every time

Bisection method



http://en.wikipedia.org/wiki/Bisection_method

Binary search the answer



need 'n' resources
what is smallest n?