CS 32

Lecture 4: Searching

"Have you tried searching under 'fruitless'?"

Naming Things

- Two main problems in computing
  - Naming things
  - Caching things
    - Pointer vs. value
  - Off-by one errors

Seen in the Wild

- `UpdateThread`
  - A function that updates a Thread
    - update:verb; thread:noun
  - A particular kind (Update) of Thread
    - update:adjective; thread:noun
  - A function that threads an Update
    - update:noun; thread:verb

First or Last?

- For a label object that will hold a date, what should we name its variable?
  - dateLabel
  - labelDate
- Note this violates previous advice

Code Completion

- If your IDE has code completion, it will start with the first letters of variable names
- Easier to use if the most generic part of the name is first
- Good for alphabetical sorts also
- Where else do we see this?

Examples

- 2017-03-13
- Homo sapiens
- Mao Zedong
- Hungarian prefix notation
- ??? Must be more

Sidebar

- `goto` Considered Harmful
  - March 1968, Dijkstra
  - Wirth changed the title
- "`goto` Considered Harmful" Considered Harmful
  - March 1987, Rubin
- ""`goto` Considered Harmful" Considered harmful" Considered Harmful

`switch` harmful?

- Back in favor
- Swift has upgraded `switch` statements
  - Pattern matching
  - `where` clauses
  - Enum-aware

Scope

- Nesting function calls
    - Tracked with a LIFO (stack)
- Locals vanish when you exit scope
- But globals are frowned upon
- What to do?

Pointers

- Everybody’s friend
  - Frenemy at best
- Pointer is “one level of indirection” away from the value
  - Need the flexibility
  - Hate the consequences

Down With Globals?

- Pointers usually point to
  - Strings
  - Images
  - Vectors or arrays
  - Instances of classes
    - aka objects

Mr Big

- Very often there are classes or data structures that are unique
- Singletons
  - Can't live with them
  - Can't live without them

Linear Search

- Example given in the text

```
while i < n && !found {
   if a[i] // is the desired item {
      found = true;
   } else {
      ++i;
   }
}
```

Average Performance

- First-pass analysis gives $(1+2+3+4+5+6+7+8+9+10) \div 10 = 5.5$
- Some assumptions there
  - HW has a handout sheet on this

Convex Sum

- Particular case of this general idea
  - $X = a_1p_1 + a_2p_2 + \cdots + a_np_n$
  - Where $\sum p_i = 1$ (aka 100%)
- A linear combination
- Also used in atomic weights

Random Value

- $X = a_1p_1 + a_2p_2 + \cdots + a_np_n$
- We say that X is a random variable
- Each time you ask it what it is, it gives a different answer!
- But there could be an average answer

Our case

- $\mathrm{E}[X] = 1 \cdot 10\% + 2 \cdot 10\% + \cdots\ 10 \cdot 10\%$
- Same as before, comes out to 5.5
- Why are we going into this convex sum stuff?
- Need it to improve the answer

Problems

- Calculation assumes each index is equally likely to be the winner
- We will let that go by
- But also... what if search fails?

Solutions

- Let's assume...
  - Search fails half the time
  - All other outcomes are equally likely
- What convex sum do we get?

Pessimistic

- $\mathrm{E}[X] = 1 \cdot 5\% + 2 \cdot 5\% + \cdots 10 \cdot 5\% + 10 \cdot 50\%$
- Comes to 7.75
- More fails $\Rightarrow$ worse performance

Better Than This?

- Could there be any kind of search more wonderful than linear search?
- Glad you asked
  - Binary
  - Hashing
  - Who knows what else?

Binary Search

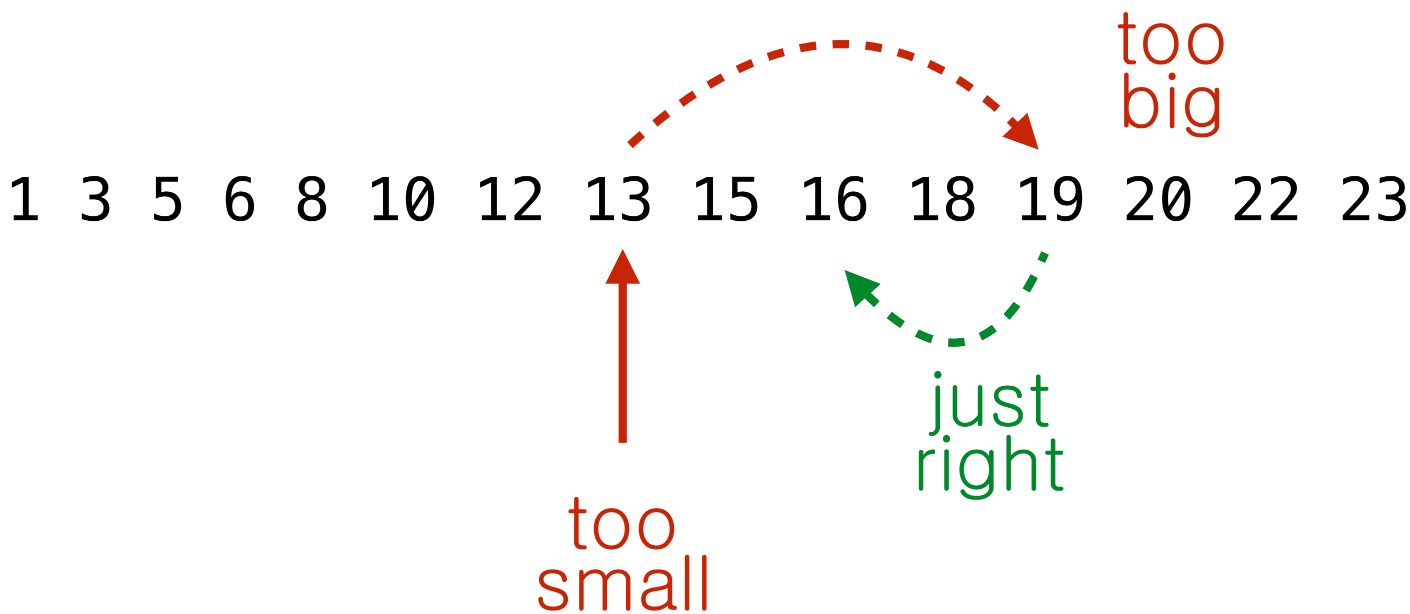
- Requires the elements be pre-sorted
- Which in turn implies a relation on the elements
- Every pair of elements must be comparable aka total ordering
- What orderings are not total?
  - Subset relations

Sub-Problems

- Start in middle
- Which side must the target be in?
- Go to the middle of that side
- Repeat until... what?

Binary In Action

Search for 16



Book Code

- The book’s code is reasonable
- So many chances for off-by-ones
- You may have found this out in the first lab

Basic Truth

- What sort of worst-case performance would you expect?
- $O(\log_2 n)$
- If you've taken CS 40, that is
- Otherwise we have to explain

Halving Function

- H(n) = the number of times you can divide n by 2 before it gets to 1
- Very closely related to $\log_2 n$
  - Basically, off by 1
- Good way to teach logarithms

STL Searches

- Operations in `<algorithm>`
- Work with `Iterators`
- We will look at binary search

Code From Somewhere

```
template< class ForwardIt, class T >
bool binary_search( ForwardIt first, ForwardIt last, const T& value );

template< class ForwardIt, class T, class Compare >
bool binary_search( ForwardIt first, ForwardIt last, const T& value, Compare comp );
```

- Uses templates
- What does that mean?

Kinda Like Overloading

```
void printMe (string s) {
   cout << "string s = \"" << s << "\"" << endl ;
}

void printMe (int i) {
   cout << "int i = " << i << endl ;
}

void printMe (string s, int i) {
   cout << s << " " << i << endl ;
}
```

- Use overloading to do different things to different types

But Different

```
template< class ForwardIt, class T >
bool binary_search( ForwardIt first, ForwardIt last, const T& value );

template< class ForwardIt, class T >
ForwardIt lower_bound( ForwardIt first, ForwardIt last, const T& value );

template< class ForwardIt, class T >
ForwardIt upper_bound( ForwardIt first, ForwardIt last, const T& value );
```

- Use templates to do the same thing to different types

STL Has It All

- Algorithms are independent of containers
- Specific versions of functions are created at build time (aka statically)
- Try to use classes, &-references and forget about pointers

Read!

- Problem Solving 8.3, 17, 18
  - Which is Templates and STL