week10

Tommy MacWilliam

tmacwilliam@cs50.net

November 14, 2011

Announcements

- pset7: returned
- quiz1: Wednesday, 11/16 during lecture
  - same locations as last time
  - course-wide review video posted online
- CS50 seminars:
  `https://manual.cs50.net/Seminars`

Today

- quiz1 review
- data structures
- bitwise operators
- HTML
- CSS
- PHP
- SQL
- JavaScript + DOM
- practice problems

Arrays

- an array is a contiguous piece of memory
- in C, arrays have a fixed size determined at declaration and consist of a single type
  - PHP and JavaScript arrays don't have these limitations
- insertion: N/A
- lookup: $O(1)$
  - address of element = index * size of each element

Linked Lists

- series of structs pointing to each other
  - elements can be inserted and removed regardless of language
- insertion: $O(n)$, but $O(1)$ possible if values are just inserted at beginning of list
- lookup: $O(n)$

```
typedef struct node {
    int value;
    struct node* next;
} node;
```

Linked Lists

- insertion
  - `malloc` space for a new node
  - make the `next` field point to what will be the next element in the list
  - make the `next` field of what will be the previous element point to the new element

Linked Lists

- iteration
  - create an iterator (pointer to a struct), which should NOT be `malloc`'d
  - make the iterator point to the root node of the list
  - after visiting a node, make the iterator point to the element specified in the `next` field
  - when `iterator == NULL`, list has been iterated over

Linked Lists

- deletion
  - use an iterator to find the node to be deleted
  - make the `next` field of the previous node point to the `next` field of the node to delete

Linked Lists

- example time!
  - `list.c`

Stacks

- when adding a new element, add to “top”
- when removing an element, take from “top”

Queues

- when adding a new element, add to “top”
- when removing an element, take from “bottom”

Stacks/Queues

- can be implemented using linked lists!
- stack: add new element to end of list, remove elements from end of list
  - or, add new element to beginning of list, remove elements from beginning of list (faster!)
- queue: add new element to end of list, remove elements from beginning of list
  - or, add new element to beginning of list, remove elements from end of list
  - speed depends on which you do more: insert or remove?

Hashtables

- hash function maps a key to a value
  - converts string into numerical index in the hashtable
- collision handling
  - chaining: each element of hashtable is a linked list of keys that hashed to that index
  - probing: store element in another (predictable) location in the table
- insertion/lookup: $O(1)$ (runtime of hash function can be considered $O(1)$)

Binary Trees

- just like linked lists, but each node can have two children instead of one

```
typedef struct btree_node {
    int value;
    struct node* left;
    struct node* right;
} btree_node;
```

Binary Trees

- invariant: condition that must be held true as elements are inserted/removed
- binary tree has two invariants
  - value of left node must be less than value of parent node
  - value of right node must be greater than value of parent node

Binary Trees

- insertion/search is best done recursively
  - just like binary search!
- compare new/search value to value of root node
  - if greater, then make right node new root node and repeat
  - if less, then make left node new root node and repeat
- insertion/lookup: $O(\log n)$

Binary Trees

- example time!
  - `binarytree.c`

Bitwise Operators

- `&`: and (both bits must be a 1)
- `|`: or (either bit can be a 1)
- `^`: xor, aka exclusive or (only one bit can be a 1)
- `>>`: right shift: move all bits over to the right
- `<<`: left shift: move all bits over to the left
- `~`: negation (flip all bits)

Bitwise Operators

- `2 & 3 = 0010 & 0011 = 0010 = 2`
- `13 | 2 = 1101 | 0010 = 1111 = 15`
- `7 ^ 3 = 0111 ^ 0011 = 0100 = 4`
- `5 << 2 = 00000101 << 2 = 00010100 = 20`
- `15 >> 2 = 00001111 >> 2 = 00000011 = 3`
- `~13 = ~1101 = 0010 = 2`

HTML

- HTML: **H**yper**T**ext **M**arkup **L**anguage
  - which is *still* not a programming language
- describes the structure and content of web pages

HTML

- page elements defined by tags
  - `<h1>welcome!</h1>`
- tags can have attributes that specify certain properties of the tag
  - `<a id=”link” class=”top” href=”http://cs50.net”>CS50</a>`

CSS

- CSS: **C**ascading **S**tyle **S**heets
- HTML describes structure and content, CSS describes aesthetics
- consists of selectors and rules
  - selector: determine what elements to style
  - rule: define what styles should be applied

CSS

- selector syntax
  - `<tag>`: select all elements with the given tag
  - `#<id>`: select all elements with the given ID
  - `.<class>`: select all elements with the given class
  - `<tag1>, <tag2>`: select all elements with either of the given tags
  - `<tag1> <tag2>`: select all elements with tag `tag2` that are descendants of `tag1`
  - `<tag>#<id>`: select all elements with the given tag and ID
  - `<tag>.class`: select all elements with the given tag and class

CSS

- 3 ways to style your page with CSS
  - create an external stylesheet, then use the `<link>` tag to add it to the page
    - example: `<link rel="stylesheet" type="text/css" href="style.css" />`
  - use the `<style>` tag
    - example: `<style> a { color: red; } </style>`
  - use the `style` attribute
    - example: `<p style="width: 300px;">`

PHP

- PHP stands for **P**HP **H**ypertext **P**rocessor
  - which is *still* cool
- PHP is server-side: code runs on the server BEFORE it is downloaded by the browser
- PHP is interpreted: no need for compiling
- PHP is dynamically-typed: no need to declare types of variables or functions, PHP figures that out for you
- PHP variables must start with a dollar sign
  - which is *still* annoying

PHP

- syntax example

```
$x = 5; $string = "hello, ";
// best function ever coming your way
function increment($number) { return ++$num
if ($x == 2)
    echo $string + "number two";
else
    echo "dr. evil";
```

SQL

- database consists of tables, which consist of columns (of different types)
- 4 types of statements (CRUD)
  - create: `INSERT`
  - read: `SELECT`
  - update: `UPDATE`
  - delete: `DELETE`

SQL

- **create:** `INSERT INTO <table> (<column1>, <column2>, ...) VALUES (<value1>, <value2>, ...)`
  - `INSERT INTO users (username, password) VALUES ('tommy', 'secret')`
- **read:** `SELECT <column1>, <column2> FROM <table> WHERE <column3> = <value3>`
  - `SELECT password FROM users WHERE username = 'tommy'`

SQL

- update: `UPDATE <table> SET <column1> = <value1>, <column2> = <value2> WHERE <column3> = <value3>`
  - `UPDATE users SET password = 'moresecret' WHERE username = 'tommy'`
- delete: `DELETE FROM <table> WHERE <column> = <value>`
  - `DELETE FROM users WHERE username = 'djm'`

PHP + SQL

- execute a SQL query: `mysql_query($query)`
  - returns a MySQL resource, not an array
- convert resource into an array: `mysql_fetch_array($resource)`
  - returns an associative array where columns are keys, values are values

JavaScript

- JavaScript is client-side: code runs on the client AFTER it is downloaded by the browser
- JavaScript is interpreted: no need for compiling
- JavaScript is dynamically-typed: no need to declare types of variables or functions, JavaScript figures that out for you
- JavaScript is *still* the best language ever
  - even if you disagree after pset8

JavaScript

- `var` keyword limits variable scope to the current function (not loop)
  - without `var`, variable is a global
- array declaration: `array = [];`
  - `array[5] = 10;`
- object/hash declaration: `object = {};`
  - `object["name"] = "tommy";`
  - `object.color = "red";`

JavaScript

- `for-in` loop can be used to iterate over an array or hash

```
for (var item in collection)
    alert(collection[item]);
```

JavaScript

- JavaScript can be used to handle page events (interactions from the user)
- events include: `onclick`, `onmouseover`, `onmouseout`, `onkeypress`, etc.
- two ways to attach event handlers to elements
  - JavaScript: get DOM object, then add property for event name
  - HTML: use attributes like `onclick` and `onmouseover` and set value equal to name of function

DOM

- DOM: **D**ocument **O**bject **M**odel
- when your browser parses a web page, it stores a representation of its structure in the global `document` object
- `document.getElementById(id)` used for retrieving an element with a specific id

DOM

- example time!
  - `events.html`

AJAX

- AJAX: **A**synchronous **J**avaScript **A**nd **X**ML
- allows JavaScript to make GET/POST requests to URLs
- functionality provided by the `XMLHttpRequest` object

AJAX

- making an AJAX request
  - create an `XMLHttpRequest` object
  - construct the URL to make the request to
  - create an event handler to handle the server response (`onreadystatechange`)
    - request is complete when HTTP status code is 200 and AJAX `readyState` is 4
  - `open` and `send` the request

AJAX

- example time!
  - `ajax.html`

Tries

- specialized form of tree
  - NOT a binary tree, since each node has more than one child
- each node corresponds to a single letter in a word
  - each child node represents the next letter in a word

```
typedef struct trie_node {
    char letter;
    int is_word;
    struct trie_node* children[27];
} trie_node;
```

Tries

- insertion/lookup
  - start at first letter of word to insert/lookup and root node of trie
  - check if child exists for the current letter
  - if so, make that child node the new root node and recurse
  - if not, create new node and make that the new root node (insert) or word is not present (lookup)
  - when no more letters in word, get/set value of `is_word` field

Practice Problems (Doug Lloyd)

- which is faster?
  - logarithmic or linear?
  - exponential or polynomial?
  - linearithmic or quadratic?

Practice Problems (Doug Lloyd)

- If we have this line in our program:
  `int array[] = {1,2,3,4};`
  What is the value of the following expression?
  `*(&(*(array+1)))`

Practice Problems

id	username	password
1	malan	bowdenfever
2	chartier	ohai
3	rbowden	bieber

- get `chartier`'s password
- change `rbowden`'s password to `heartthrob`
- delete `malan` from the database

Practice Problems

- 2009: 12, 15, 23, 26-27
- 2008: 1-3, 5, 15