CS 105

Week 5

Midterm #1

Tuesday, Oct. 7, 2014

8:00pm – 9:30pm

Midterm #1 Conflict Exam Signup

Posted on course website tonight!

Deadline: Tuesday by 9:00pm

CS 105: Improved

9am Lecture Video Recorded

TA Lecture Notes

Lecture Review

Which conditional does **not** contain an error in the code?

A) `if (a = 50)`

B) `if (x =! 40)`

C) `if (c > 4 || c < 2)`

D) `if (x > 20 & > 30)`

E) `IF (y <= 50)`

Which conditional does not contain an error in the code?

A) `if (a = 50)`

B) `if (x =! 40)`

C) `if (c > 4 || c < 2)`

D) `if (x > 20 & > 30)`

E) `IF (y <= 50)`

```
var a =
  [ [2, 3], [4, 5], [6, 2] ];
```

What is the value of `a[1][1]`?

A) 2

B) 3

C) 4

D) 5

E) 6

```
var a =
  [ [2, 3], [4, 5], [6, 2] ];
```

What is the value of `a[1][1]`?

A) 2

B) 3

C) 4

D) 5

E) 6

Searching

How do I search for a string inside an array?

```
var s = "carrot";
var fruits = ["apple", "banana",
              "grape", "orange",
              "strawberry“];
```

How do I search for a string inside an array?

```
var s = "carrot";
var fruits = ["apple", "banana",
              "grape", "orange",
              "strawberry"];

// for each element in the array:
//   check if the element is a match
```

```
var s = "carrot";
var fruits = ["apple", "banana",
              "grape", "orange",
              "strawberry“];

for (var i = 0; i < fruits.length;
     i++)
{
   if (s == fruits[i])
   {
      return true;
   }
}
return false;
```

```
function linearSearch(s, list)
{
   for (var i = 0; i < list.length;
         i++)
   {
      if (s == list[i])
      {
         return true;
      }
   }
   return false;
}
```

Linear Search

- Move sequentially through an array, checking each element for a match.

- If the length of the array is…
  - 100:

Linear Search

- Move sequentially through an array, checking each element for a match.

- If the length of the array is…
  - 100: check up to 100 elements.

Linear Search

- Move sequentially through an array, checking each element for a match.

- If the length of the array is…
  - 100: check up to 100 elements.
  - 1000:

Linear Search

- Move sequentially through an array, checking each element for a match.

- If the length of the array is…
  - 100: check up to 100 elements.
  - 1000: check up to 1000 elements.

Linear Search

- Move sequentially through an array, checking each element for a match.

- If the length of the array is…
  - 100: check up to 100 elements.
  - 1000: check up to 1000 elements.

Linear Search

- Move sequentially through an array, checking each element for a match.

- If the length of the array is…
  - 100: check up to 100 elements.
  - 1000: check up to 1000 elements.

- *If the list doubles, it takes twice as long.*

Better Searching?

Binary Search

- Look at an element in the center of an array. Remove the half of the array where the element would not exist.

"apple"

"banana"

"blackberry"

"blueberry"

"grape"

"kiwi"

"lemon"

"orange"

"strawberry"

"carrot"

“apple”

“banana”

“blackberry”

“blueberry”

“grape”

“kiwi”

“lemon”

“orange”

“strawberry”

“carrot”

"apple"

"banana"

"blackberry"

"blueberry"

"grape"

"kiwi"

"lemon"

"orange"

"strawberry"

"carrot"

“apple”

“banana”

“blackberry”

“blueberry”

“grape”

“kiwi”

“lemon”

“orange”

“strawberry”

“carrot”

"apple"

"banana"

"blackberry"

"blueberry"

"grape"

"kiwi"

"lemon"

"orange"

"strawberry"

"carrot"

"apple"

"banana"

"blackberry"

"blueberry"

"grape"

"kiwi"

"lemon"

"orange"

"strawberry"

"carrot"

"apple"

"banana"

"blackberry"

"blueberry"

"grape"

"kiwi"

"lemon"

"orange"

"strawberry"

"carrot"

Binary Search

- Look at an element in the center of an array. Remove the half of the array where the element would not exist.

- If the length of the array is…
  - 9:

Binary Search

- Look at an element in the center of an array. Remove the half of the array where the element would not exist.

- If the length of the array is…
  - 9: up to 3 comparisons

Binary Search

- Look at an element in the center of an array. Remove the half of the array where the element would not exist.

- If the length of the array is…
  - 9: up to 3 comparisons
  - 18:

Binary Search

- Look at an element in the center of an array. Remove the half of the array where the element would not exist.

- If the length of the array is…
  - 9: up to 3 comparisons
  - 18: up to 4 comparisons

Binary Search

- If the length of the array is…
  - 9: up to 3 comparisons
  - 18: up to 4 comparisons
  - 100: up to 7 comparisons

Binary Search

- If the length of the array is…
  - 9: up to 3 comparisons
  - 18: up to 4 comparisons
  - 100: up to 7 comparisons
  - 1000: up to 10 comparisons

Binary Search

- If the length of the array is…
  - 9: up to 3 comparisons
  - 18: up to 4 comparisons
  - 100: up to 7 comparisons
  - 1000: up to 10 comparisons
  - 1,000,000: up to 20 comparisons

Binary Search

- If the length of the array is…
  - 9: up to 3 comparisons
  - 18: up to 4 comparisons
  - 100: up to 7 comparisons
  - 1000: up to 10 comparisons
  - 1,000,000: up to 20 comparisons
  - 7,000,000,000: only 33 comparisons!

Binary Search

- Look at an element in the center of an array. Remove the half of the array where the element would not exist.

Binary Search

- Look at an element in the center of a **sorted** array. Remove the half of the array where the element would not exist.

Binary Search

- Look at an element in the center of a sorted array. Remove the half of the array where the element would not exist.

- *If you double the size of the data, only one additional comparison is needed.*
  - *Allows for fast processing of big data.*

Sorting

Sorting vs. Searching

- Searching is “easy”:
  - Unsorted → Linear Search
  - Sorted → Binary Search
- Sorting is “hard”:
  - Hundreds of different algorithms

Selection Sort

- Find the element that is alphabetically first in the list.
- Swap that element with the first element in the list.
- Repeat for the 2nd element.

“lemon”

“grape”

“apple”

“strawberry”

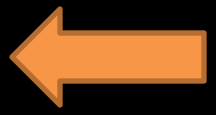
“banana”

“kiwi”

“blackberry”

“orange”

“blueberry”



"lemon"

"grape"

"apple"

"strawberry"

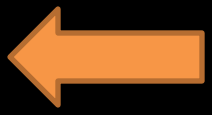
"banana"

"kiwi"

"blackberry"

"orange"

"blueberry"



"lemon"

"grape"

"apple"

"strawberry"

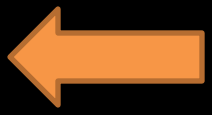
"banana"

"kiwi"

"blackberry"

"orange"

"blueberry"



"lemon"

"grape"

"apple"

"strawberry"

"banana"

"kiwi"

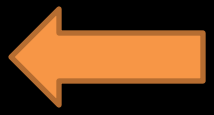
"blackberry"

"orange"

"blueberry"

“lemon”

“grape”

“apple” 

“strawberry”

“banana”

“kiwi”

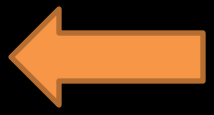
“blackberry”

“orange”

“blueberry”

"lemon"

"grape"

"apple" 

"strawberry"

"banana"

"kiwi"

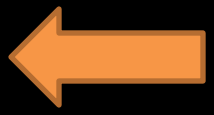
"blackberry"

"orange"

"blueberry"

"lemon"

"grape"

"apple" 

"strawberry"

"banana"

"kiwi"

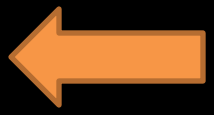
"blackberry"

"orange"

"blueberry"

"lemon"

"grape"

"apple" 

"strawberry"

"banana"

"kiwi"

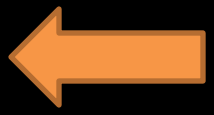
"blackberry"

"orange"

"blueberry"

"lemon"

"grape"

"apple" 

"strawberry"

"banana"

"kiwi"

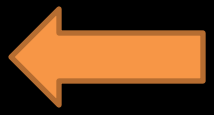
"blackberry"

"orange"

"blueberry"

"lemon"

"grape"

"apple" 

"strawberry"

"banana"

"kiwi"

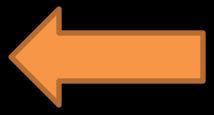
"blackberry"

"orange"

"blueberry"

"lemon"

"grape"

"apple" 

"strawberry"

"banana"

"kiwi"

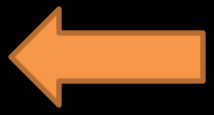
"blackberry"

"orange"

"blueberry"

“lemon”

“grape”

“apple” 

“strawberry”

“banana”

“kiwi”

“blackberry”

“orange”

“blueberry”

“lemon”

“grape”

“apple”

“strawberry”

“banana”

“kiwi”

“blackberry”

“orange”

“blueberry”

“apple”

“grape”

“lemon”

“strawberry”

“banana”

“kiwi”

“blackberry”

“orange”

“blueberry”

Popular Sorting Algorithms

- Selection Sort
- Bubble Sort
- Merge Sort
- Quick Sort
- Radix Sort