THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

COMP 110
Introduction to Programming

Fall 2015

Time: TR 9:30 – 10:45

Room: AR 121 (Hanes Art Center)

Jay Aikat

aikat@cs.unc.edu

Previous Class

- What did we discuss?

Today

- Nested loops and methods review session today in SN 014: 7-9 PM
- Assignment 4: Part A due on Tue, 11/17
- Today – Sorting

Sorting

- Think about your own field – Biology, Psychology, Math, Business…
- Can you think of examples when you need to search and sort data?
- How do you do it?

Algorithms

- All those applications you use for sorting data or searching for a specific piece of information in a large database use **algorithms**!

- Anyone use Google? ☺

Sorting - example

- Given an array of numbers, sort it into ascending/descending order
- Before sorting:

4	7	3	9	6	2	8

- After sorting:

2	3	4	6	7	8	9

Searching Arrays

- Searching arrays for a particular value
- Sorting arrays
  - makes searching for a particular value easier (and quicker)

Searching Arrays

- Find one particular element in an array of many elements
- Find several particular elements in an array of many elements
- Complexity (How Long To Search?)
  - find a parking space – *linear (go down the line…)*
  - look up a word in a dictionary - *complex*
    - 500K+ words in OED
  - Web search - *very complex*
    - Trillions of web pages

Searching Arrays

- Linear search
  - it's like looking for a parking space
- Binary search
  - (sort of like) searching for a word in the dictionary

Linear Search

- Check the first item, then the second item, and so on… until?
  - you find the target item, OR
  - you reach the end of the list
- That is **linear** (or sequential) search

Linear Searching

- Given a target value and an array of integers
  - the array list can be sorted (or not)
  - walk through the array
    - repeatedly ask: Is this a match?
    - quit when the answer is yes (use break stmt)
  - if you reach the end of the array, there is no match
- Inefficient
  - worst time to search is ~ length
  - average time to search is ~ length/2
- Relatively easy to program

Linear Search - Integers

```
// Linear search of unordered list of integers

int[] list = {17, 14, 9, 23, 18, 11, 62, 47, 33, 88}; // unordered list

int searchFor = 33; // look for this value in the list

// Loop thru list until we find match

int foundAt = -1; // where found (default)

        for (int index = 0; index < list.length; index++) {

                if (list[index] == searchFor) {
                        foundAt = index;
                        break;          // jump out of the loop
                }
        }
// foundAt is now index of item "searchFor" or -1 if not found
```

Linear Search - Strings

```
// Linear search of unordered list of Strings

// unordered list
String[] list = {"Bart", "Homer", "Marge", "Lisa", "Maggie", "Millhouse"};

String searchFor = "Maggie"; // look for this value in the list

// Loop thru list until we find match
int foundAt = -1; // where found (default)

        for (int index = 0; index < list.length; index++) {

                if (list[index].equals(searchFor)) {
                        foundAt = index;
                        break;          // jump out of the loop
                }
        }
// foundAt is now index of item "searchFor" or -1 if not found
```


Binary Search

- How would you search a word in a dictionary? E.g. "Spring" ?
- You look in the second half of the dictionary!
- Dictionary is sorted…
- We already use **binary search**!

Binary Search

- Requires ordered (sorted) list
- Set a searchRange – begin with the entire list
- Repeat:
  - pick a "test value" in the middle of searchRange
  - if test value == value searching for
    - Stop!
  - if test value > value searching for
    - searchRange = lower half of searchRange
  - if test value < value searching for
    - searchRange = upper half of searchRange

Binary Search - Example

Looking for 46

Trial 1

2 4 5 12 16 19 22 26 29 32 37 41 46 50

2

2 4 5 12 16 19 22 26 29 32 37 41 46 50

3

2 4 5 12 16 19 22 26 29 32 37 41 46 50

Notes on Binary Searches

- List must be ordered (sorted)
- Much more efficient than linear search
  - in example, took 3 iterations instead of 13 for linear
  - linear
    - worst case ~ listLength
    - average ~ listLength/2
  - for 100K words: 17 iterations versus 50,000
- More complex to program

Question (take 2 min; do it yourself)

2 10 17 45 49 55 68 85 92 98

How many comparisons are needed to determine if the following items are in the list of 10 items?

number	linear search	binary search
15	10 (3, if know list sorted)	3 (49, 10, 17)
49	5	1
98	10	4 (49, 85, 92, 98)
2	1	3 (49, 10, 2)

Sorting

- Put elements of an array in some order
  - alphabetize names
  - order grades lowest to highest
- Two simple sorting algorithms
  - selection sort
  - insertion sort

Selection Sort

- Sorts by putting values directly into their final, sorted position
- For each value in the list, the selection sort finds the value that belongs in that position and puts it there

Selection Sort

- Scan the list to find the smallest value
- Exchange (swap) that value with the value in the first position in the list
- Scan rest of list for the next smallest value
- Exchange that value with the value in the second position in the list
- And so on, until you get to the end of the list

Selection Sort at work

98 68 83 74 93

68 98 83 74 93

68 **74** 83 98 93

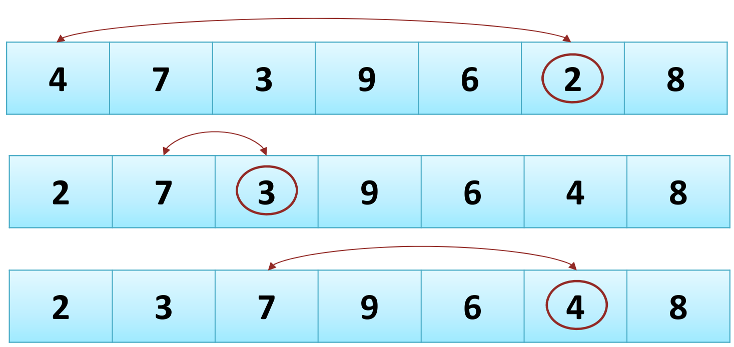
68 **74** **83** 98 93

68 **74** **83** **93** **98** **SORTED!**

Selection Sort

- Sorts in ascending order
- Can be changed to sort in descending order
  - look for max instead of min

Selection Sort – another example



and so on…

Swap

```
private static void swap(int i, int j, int[] a) {
      int temp = a[i];
      a[j] = a[j];
      a[j] = temp;
}
```

- This method will swap the value of a[i] and a[j]

Demo

http://www.sorting-algorithms.com/

Next class

- More Sorting!

COMP 110 - Fall 2015 27