CSSE 220 Day 25

VectorGraphics Demonstrations
Sorting
Searching
Mini-project teams

No new checkout from SVN

Exam2 question 7

- Question on VectorGraphics project
- 5 minutes

Project demos

- At most 5 minutes per team
- Hand me your script after you are done.

Questions?

- Exam
- MineSweeper
- Anything else

- Day 25 HW is due Wednesday
  - A sorting Exercise
  - Some problems to look at (and be sure you can do them sometime before the Final exam)
  - Start-up for MineSweeper
    - Lists of classes and responsibilities, CRC cards, user stories for development cycle #1

What is sorting?

- Let’s see…

Course Goals for Sorting: You should…

- Be able to describe basic sorting algorithms:
  - Selection sort
  - Insertion sort
  - Merge sort
- Know the run-time efficiency of each
- Know the best and worst case inputs for each

Selection Sort

- Basic idea:
  - Think of the list as having a sorted part (at the beginning) and an unsorted part (the rest)
  - Find the smallest number in the unsorted part
  - Exchange it with the element at the beginning of the unsorted part (making the sorted part bigger and the unsorted part smaller)

Repeat until unsorted part is empty

Profiling Selection Sort

- Profiling: collecting data on the run-time behavior of an algorithm
- How long does selection sort take on:
  - 10,000 elements?
  - 20,000 elements?
  - 40, 000 elements?
  - 80,000 elements?
  - . . .

Q4

Analyzing Selection Sort

- Analyzing: calculating the performance of an algorithm by studying how it works, typically mathematically
- Typically we want the relative performance as a function of input size

- Example: For an array of length *n*, how many times does **`selectionSort()`** call **`compareTo()`**?

Handy Fact

$$1 + 2 + \ldots + (n-1) + n = \frac{n(n+1)}{2}$$

Q5–10

Insertion Sort

- Basic idea:
  - Think of the list as having a sorted part (at the beginning) and an unsorted part (the rest)

  - Get the first number in the unsorted part
  - Insert it into the correct location in the sorted part, moving larger values up in the array to make room

Repeat until unsorted part is empty

Insertion Sort Exercise

- Profile insertion sort
- Analyze insertion sort assuming the inner *while* loop always runs the maximum possible number of times
- What input order causes the worst case behavior? The best case?
- Does the input order affect selection sort?

Do for Homework!

Q11-20

Searching

- Consider:
  - Find the CRN of CSSE220-01 in the next term's printed schedule
  - Find the course whose CRN is 3099
- Why is one task harder than the other?
- For searching unsorted data, what’s the worst case number of comparisons we would have to make?
- How to sequentially search sorted data?
  - See Session 22 slides, where we also talked about its running time.

Binary Search of Sorted Data

- A divide and conquer strategy
- Basic idea:
  - Divide the list in half
  - Decide whether result should be in upper or lower half
  - Recursively search that half

Analyzing Binary Search

- What's the best case?
- What's the worst case?
- We use recurrence relations to analyze recursive algorithms:
  - Let T(*n*) count the number of comparisons to search an array of size *n*
  - Examine code to find recursive formula of T(*n*)
  - Solve for *n*

Code: Binary Search

```
public static final int NOT_FOUND = -1;
public static <T extends Comparable<? super T>>
   int binarySearch( T[ ] a, T x ) {
       int low = 0;
       int high = a.length - 1;
       int mid;
       while( low <= high ) {
           mid = ( low + high ) / 2;

           if( a[ mid ].compareTo( x ) < 0 )
               low = mid + 1;
           else if( a[ mid ].compareTo( x ) > 0 )
               high = mid - 1;
           else
               return mid;
       }
       return NOT_FOUND;     // NOT_FOUND = -1
   }
```

Interpolation search

- A more natural appraoch.
- If you were looking for my name in the phone book, would you start your search in the middle?
- In interpolation search, we choose where in the table to probe based on the value of the key relative to the first and last keys in the part of the table we are searching.

Interpolation search

- general formula: when looking for item x in a[low] … a[high], the next place to search is:

$$next = low + \left\lfloor \frac{x - a[low]}{a[high] - a[low]} * (high - low) \right\rfloor$$

- Average case # of probes:
- Simple references:
  - Weiss* Section 5.6.3
  - http://en.wikipedia.org/wiki/Interpolation_search

* Data Structures and Problem Solving in Java, 3rd Edition. (CSSE 230 textbook, on reserve in the library, under either 220 or 230 - anderson)

Interpolation search limitation

- What if the data is not uniform?
- Phone book
- Phone book in Wilkes-Barre, PA
- RHIT CSSE staff members, 1986-2009.

RHIT CSSE staff members, 1986–2009 (all that I can remember)

anderson
atkins
ardis
azhar
bagert
baker
bohner
boutell
bowman
chenoweth
chidanandon

clifton
criss
curry
dalkolic
defoe
degler
jeschke
kaczmzrczyk
kinley
laxer

mohan
mellor
merkle
mutchler
oexmann
sengupta
surendran
sullivan
wollowski
young

The downsides of binary search and interpolation search

- Initially sorting the array (expensive)
- Keeping it sorted if the data changes
    - That's why we call these techniques "static search"
    - Other approaches (such as linked lists, trees, and hash tables) work better for dynamic data

MineSweeper project

- Same teams as VectorGraphics
  - By Sunday afternoon, no one had
    - filled out a new preference saying they didn't want to work with any of their current teammates, or
    - filled out partner evaluation that gave any partner a bad evaluation.
  - So I decided to leave the teams the same.
- Same approach:
  - First: class list, responsibility list, CRC cards, UML diagram
  - Four development cycles, with user stories for each.