CSSE 220 Day 26

Continue the Sorting intro
Work on Spellchecker Project

CSSE 220 Day 26

- Turn in written problem now.
- If you find a good dictionary to use, please post a link to it on the Mini-project discussion forum.
- Everything for the Mini-project is due at the beginning of your class time on Day 30. No late days may be used for this one.
  - Why?
  - Presentations in class that day
  - Graders are students, too.
- There will be time in class to work with your team every day. Do not miss it!

Project presentation/demonstration

- Day 30 in class
- Informal and informational
- What does your program do? How does it do it
- Data Structures and algorithms.
- Intended audience: Your classmates
  - Already know what the project is.
  - Already know Java'
  - Already know the data structures involved.
- No more than 7 minutes, including Q&A time.

Today's Agenda

- Work on Spellchecker
- Continue the Sorting intro

Project work

- Before you leave today:
  - UML Class Diagram
  - Iterative enhancement plan
  - Commit to your repository
- Finish UML diagram and iterative enhancement plan before midnight tonight.

THE DEPARTMENT OF COMPUTER SCIENCE & SOFTWARE ENGINEERING

INVITES YOU TO THE

DIRECTOR OF SOFTWARE ENGINEERING FACULTY CANDIDATE TALK

SHAWN BOHNER
VIRGINIA TECH

SOFTWARE SYSTEMS CHANGE TOLERANCE: AN EVOLVING PERSPECTIVE

FRIDAY FEBRUARY 8, 2008 4:30 P.M
O269

Homework

- Finish UML Class Diagram and IEP today
- Markov partner evaluation survey
- Two written problems
- Substantial progress on SpellChecker

IntegerPower Solutions

```
public static double integerPower(double x, int n){
   if (n < 0 )
     throw new IllegalArgumentException("negative power");
   double prod=1, power = x;
   while (n > 0) {
      if (n % 2 == 1)
         prod *= power;
      power = power*power;
      n = n / 2;
   }
   return prod;
}
```

Simple recursive solution:

```
public static double integerPower(double  x, int n){
   if (n == 0)
      return 1;
   if( n%2 == 0)
      return integerPower(x*x, n/2);
   return x*integerPower(x*x, n/2);
}
```

Sorting Intro

- What do we mean by "sort"?
- What is the best sorting algorithm?
- The three very simple Algorithms
  - Bubble Sort
    - Why is it so slow?
  - Insertion sort
  - Selection sort
- Inversions and movement
- Faster algorithms

Knowledge of Elementary Sorts

- What should you know/be able to do by the end of this course?
  - The basic idea of how each sort works
    - insertion, selection, bubble, shell, merge
  - Can write the code in a few minutes
    - insertion, bubble, selection
    - perhaps with a minor error or two
    - not because you memorized it, but because you understand it
  - What are the best case and worst case orderings of N data items? For each of these:
    - Number of comparisons
    - Number of data movements

Insertion sort

- for (i=1; i< N; i++)
  - place a[i] in its correct position relative to a[0] …a[i-1]
    - to do this, we need to move "right" each of those items that is smaller than a[i].

- We wrote the code in yesterday's class
- Number of data comparisons, movements:
  - best case
  - worst case

Selection sort

- Find largest element and exchange it with the last element in the array
- Find second largest element and exchange it with the next-to-last element in the array
- etc.
- Code
- Comparisons and Data movements
- Best case, Worst Case

Bubble Sort

- Basic idea
- Code
- Number of comparisons, data movements.
  - Best case
  - Worst case
  - Inversions
- Proposed improvement: two-way bubble sort
- Demonstrations:
  - http://www.cs.ubc.ca/~harrison/Java/sorting-demo.html
  - http://www.geocities.com/siliconvalley/network/1854/Sort1.html

Shell sort

- 1959, Donald Shell
- Based on insertion sort
- Faster because it compares elements with a gap of several positions
- For example, if the gap size is 8,
  - Insertion sort elements 0, 8, 16, 24, 32, 40, …
  - Insertion sort elements 1, 9, 17, 25, 33, 41, …
  - …
  - Insertion sort elements 7, 15, 23, 31, 39, 47, …
- Elements that are far out of order are quickly moved closer to where they are supposed to go.

Shell sort gap sizes

- Start with a large gap
- Do it again with a smaller gap
- Keep decreasing the gap size
- The last time, the gap must be 1 (why?)
- No gap size should be a multiple of another (except all are multiples of 1)
- $O(n (\log n)^2)$

Shellsort animation

- http://www.cs.princeton.edu/~rs/shell/animate.html

Merge Sort

- Divide and conquer
- Sort each half, merge halves together
- How to sort each half?