

## Clip Art Sources

- ▲ [www.barrysclipart.com](http://www.barrysclipart.com)
- ▲ [www.livinggraphics.com](http://www.livinggraphics.com)
- ▲ [www.rad.kumc.edu](http://www.rad.kumc.edu)
- ▲ [www.graphicmaps.com](http://www.graphicmaps.com)


## What The Course Is About




- ▲ Data structures is concerned with the representation and manipulation of data.
- ▲ All programs manipulate data.
- ▲ So, all programs represent data in some way.
- ▲ Data manipulation requires an algorithm.

## What The Course Is About



- Algorithm design methods needed to develop programs that do the data manipulation.
- The study of data structures and  algorithms is fundamental to Computer Science.



## Prerequisites

- ▲ Java 
- ▲ Asymptotic Complexity
  - Big Oh, Theta, and Omega notations



## Web Site



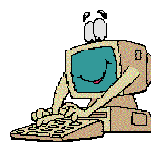
- ▲ [www.cise.ufl.edu/~sahni/cop3530](http://www.cise.ufl.edu/~sahni/cop3530)
- ▲ Handouts, syllabus, text, source codes, exercise solutions, lectures, assignments, past exams, past exam solutions, TAs, etc. 
- ▲ My office data. 



## Assignments



- ▲ Assignment guidelines
- ▲ Submission procedures
- ▲ Do Assignment 0 by next week.



## Source Codes

- ▲ Read download and use instructions.
- ▲ Must have Java 1.2 or higher.
- ▲ See ProgramIndex.htm, AllNames.html and other html files produced by Javadoc for Java codes.



## Discussion Sections



- ▲ Go to any one
- ▲ TA will answer your questions
- ▲ TA will go through a few exercises from the book
- ▲ Web site lists what is done in each meeting of the discussion section



## Organization of Text



- ▲ Three parts
- ▲ Part I ... Chapters 1-4, Background
- ▲ Part 2 ... Chapters 5-17, Data Structures
- ▲ Part 3 ... Chapters 18-22, Algorithms
- ▲ Each chapter ... concepts + applications

## Grades

- ▲ 25% for assignments
- ▲ 25% for each test

## Grades (Rough Cutoffs)

- ▲ A  $\geq 83\%$
- ▲ B+  $\geq 75\%$
- ▲ B  $\geq 70\%$
- ▲ C+  $\geq 65\%$
- ▲ C  $\geq 60\%$
- ▲ D+  $\geq 55\%$
- ▲ D  $\geq 50\%$

## Sorting

- ▲ Rearrange  $a[0], a[1], \dots, a[n-1]$  into ascending order. When done,  $a[0] \leq a[1] \leq \dots \leq a[n-1]$
- ▲  $8, 6, 9, 4, 3 \Rightarrow 3, 4, 6, 8, 9$

## Sort Methods

- ▲ Insertion Sort
- ▲ Bubble Sort
- ▲ Selection Sort
- ▲ Count Sort
- ▲ Shaker Sort
- ▲ Shell Sort
- ▲ Heap Sort
- ▲ Merge Sort
- ▲ Quick Sort

## Insert An Element

- ▲ Given a sorted list/sequence, insert a new element
- ▲ Given 3, 6, 9, 14
- ▲ Insert 5
- ▲ Result 3, 5, 6, 9, 14

## Insert an Element

- ▲ 3, 6, 9, 14    insert 5
- ▲ Compare new element (5) and last one (14)
- ▲ Shift 14 right to get 3, 6, 9, , 14
- ▲ Shift 9 right to get 3, 6, , 9, 14
- ▲ Shift 6 right to get 3, , 6, 9, 14
- ▲ Insert 5 to get 3, 5, 6, 9, 14



## Insert An Element

```
// insert t into a[0:i-1]
int j;
for (j = i - 1; j >= 0 && t < a[j]; j--)
    a[j + 1] = a[j];
a[j + 1] = t;
```

## Insertion Sort

- ▲ Start with a sequence of size 1
- ▲ Repeatedly insert remaining elements

## Insertion Sort

- ▲ Sort 7, 3, 5, 6, 1
- ▲ Start with 7 and insert 3 => 3, 7
- ▲ Insert 5 => 3, 5, 7
- ▲ Insert 6 => 3, 5, 6, 7
- ▲ Insert 1 => 1, 3, 5, 6, 7

## Insertion Sort

```
for (int i = 1; i < a.length; i++)  
{// insert a[i] into a[0:i-1]  
    // code to insert comes here  
}
```

## Insertion Sort

```
for (int i = 1; i < a.length; i++)  
{// insert a[i] into a[0:i-1]  
    int t = a[i];  
    int j;  
    for (j = i - 1; j >= 0 && t < a[j]; j--)  
        a[j + 1] = a[j];  
    a[j + 1] = t;  
}
```