Welcome to CSE 373: Data Structures & Algorithms

✦ Instructor: Rajesh Rao (rao@cs.washington.edu)
✦ TAs:
  ⇒ Charles Gordon (cgordon@cs)
  ⇒ Jiwon Kim (jwkim@cs)
✦ Class web page for syllabus and course information:
  ⇒ http://www.cs.washington.edu/education/courses/373/01sp/
✦ Add yourself to the mailing list → see the web page
✦ Textbook
  ⇒ Data Structures and Algorithm Analysis in C
     by Mark Allen Weiss (2nd ed, 1997)

Today’s Lecture

✦ Course Topics
✦ Course Goals
✦ Overview of Selected Topics from Chapter 1
  ⇒ Recursion
  ⇒ Proof by Induction
✦ Class Survey

Course Topics

✦ Mathematical Preliminaries (Chap. 1)
✦ Introduction to Algorithm Analysis (Chap. 2)
✦ Review of Lists, Stacks, and Queues (Chap. 3)
✦ Varieties of Trees and Search Algorithms (Chap. 4)
✦ Hashing and Heaps (Chaps. 5 & 6)
✦ Sorting out various Sorting Algorithms (Chap. 7)
✦ Disjoint Sets and Union-Find (Chap. 8)
✦ Graph Algorithms (Chap. 9)

Grading, Homework, and other logistics

✦ Weekly homework assignments (50%)
  ⇒ Approximately 5 written and 3 programming assignments
  ⇒ No late submissions
  ⇒ However, lowest score will be dropped
✦ Midterm exam (25%)
  ⇒ Monday, April 30, 2001
✦ Final (25%)
  ⇒ Wednesday, June 6, 2001
Data Structures (DS): What, How, and Why?

✦ Programs receive, manipulate, and output data
✦ Need to organize data according to problem being solved
   ◦ Data structures are methods for organizing data
✦ Formal definition of DS: Abstract Data Type (ADT) - A “toolkit” of operations for manipulating data
   ◦ E.g. A list defined using struct with operations insert and delete
✦ Program design depends crucially on data organization i.e. how data is structured for use by the program
   ◦ Implementation of some operations may become easier or harder
   ◦ Speed of program may dramatically decrease or increase
   ◦ Memory used may increase or decrease
   ◦ Debugging may be become easier or harder

Course Goals for Data Structures

✦ Study different implementation techniques for some fundamental ADTs
✦ Learn how to choose the “best” one
✦ Learn how to modify standard ADTs for specific problems, and create new ADTs

Analysis of Algorithms

✦ What is an algorithm?
   ◦ A sequence of steps (a “program”) that accomplishes a task
✦ Many different algorithms may correctly solve a given task
✦ But choice of a particular algorithm may have enormous impact on time and memory used
   ◦ Time versus space tradeoffs are very common
✦ Choice of algorithm and choice of data structure for a task are often interdependent

Course Goals for Algorithm Analysis

✦ Understand the mathematical fundamentals needed to analyze algorithms
✦ Learn how to compare the efficiency of different algorithms in terms of running time and memory usage
✦ Study a number of standard algorithms for data manipulation and learn to use them for solving new problems
A Simple Example for Today’s Class

Problem: Find the sum of the first num integers stored in an array v.

```c
int sum ( int v[ ], int num)
{
    int temp_sum, i;
    temp_sum = 0;
    for ( ? )
        temp_sum = ? ;
    return temp_sum;
}
```

A Simple Example (cont.)

Problem: Find the sum of the first num integers stored in array v.

```c
int sum ( int v[ ], int num)
{
    int temp_sum, i;
    temp_sum = 0;
    for ( i = 0; i < num; i++ )
        temp_sum = temp_sum + v[i] ;
    return temp_sum;
}
```

Programming via Recursion

New Twist: Write a recursive function to find the sum of the first num integers stored in array v.

```c
int sum ( int v[ ], int num)
{
    ?
}
```

Programming via Recursion

New Twist: Write a recursive function to find the sum of the first num integers stored in array v.

```c
int sum ( int v[ ], int num)
{
    if (num == 0) return 0;
    else return v[num-1] + sum(v,num-1);
}
```

- Simplifies the code dramatically
- Is the program correct?
  - Proof by induction
Proof of Program Correctness by Induction

```c
int sum ( int v[ ], int num)
{ 
    if (num == 0) return 0;
    else return v[num-1] + sum(v,num-1); }
```

Need to prove: \( \text{sum}(v,n) \) correctly returns the sum of the first \( n \) elements of array \( v \), for any \( n \geq 0 \).

**Basis Step:** Program is correct for \( n=0 \): Returns 0 ✔

**Inductive Hypothesis (\( n=k \)):** Assume \( \text{sum}(v,k) \) returns sum of first \( k \) elements of \( v \), i.e. \( v[0]+v[1]+\ldots+v[k-1] \)

**Inductive Step (\( n=k+1 \)):** \( \text{sum}(v,k+1) \) returns: \( v[k] + \text{sum}(v,k) \) which is the sum of the first \( k+1 \) elements of \( v \). ✔

Next Class: Analysis of Algorithms

✦ Things to do this week:
  ➤ Visit course website
  ➤ Sign up for mailing list (instructions on class website)
  ➤ Find the MSCE lab and make sure you can run Visual C++
  ➤ Read Chapters 1 and 2

✦ Adds: There have been more requests for add codes than the number of slots available. Send e-mail to Crystal Eney (ceney@cs) explaining why you need to take this course this quarter.

✦ Please complete and hand-in the class survey before you leave!