Welcome to CSE 326!
Data Structures

Pick up…
- First day survey
- Copy of lecture slides
- Textbook errata
- Course syllabus

1

Today's Outline

- Administrative Info
- Survey
- Overview of the Course
- What is an algorithm? ADT? Data structure?
- Review: Stacks and queues

2

Who am I?

<u>Ashish</u> Sabharwal

n^{th} year grad student, CSE [currently n = 6]
Please *don't* call me "professor"… yet!

- Research — Theory: Algorithms, complexity
  Applications: Solvers for AI, model checking, etc.
- Teaching — TA'ed several courses
  My first full class!

3

What's this tabletPC thing I'm walking around with?

Classroom Presenter system
- Richard Anderson, Steve Wolfman, et al.

Allows cool stuff such as
- Presenter and viewer modes
- Writing on slides

Still under construction.
Will it fail? I don't know…

4

Course Staff and Textbook

- **Instructor**: Ashish Sabharwal, Allen Center 214, *ashish@cs*
  Office hours: TBD
- **Teaching Assistants**:

Ethan Phelps-Goodman	*ethanpg@cs*	Sections, concepts
Albert J. Wong	*awong@cs*	Programming guru, special tutorials (eg. unix)

  Office hours: TBD
- **Textbook**: *Data Structures & Algorithm Analysis in Java*
  - by Mark Allen Weiss

Course Mechanics

- Web page: `http://www.cs.washington.edu/326`
- Mailing aliases
  - announcement list *cse326-announce@cs*
  - discussion list *cse326@cs*
  - staff alias *cse326-staff@cs*
  - <u>subscribe to the lists</u> using web interface; see webpage
- Course laboratories are 002, 006, 022 Allen Center
  - labs have NT machines with X servers to access UNIX
  - All programming projects graded on UNIX server <u>attu.cs</u>
  - OK to develop using other tools (e.g. under Windows) but make sure you test under UNIX

6

Course Policies

- Written assignments
  - Due at the start of class on due date; late homeworks not accepted!
- Programming assignments
  - Turned in electronically before 11pm on due date
  - Once per quarter: use your “late day” for extra 24 hours – **Must email TA**
- Work in teams only on explicit team projects
  - Appropriate *discussions* encouraged – see website
- Approximate Grading

– Assignments:	30%	
– Midterm:	20%	Monday Nov 3, in class
– Final:	30%	Monday Dec 15, in class (2 hours)
– **Best of above 3:**	10%	
– Participation & quizzes:	10%	

7

A quick break before we delve into course material!

- Fill out the survey
- Tell me times that are BAD for office hours

8

Today’s Outline

- Administrative Info
- Survey
- Overview of the Course
- What is an algorithm? ADT? Data structure?
- Review: Stacks and queues

9

What is this Course About?

Clever ways to organize information in order to enable efficient computation

- What do we mean by clever?
- What do we mean by efficient?

10

Clever? Efficient?

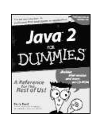
Data Structures	Algorithms
Lists, Stacks, Queues	Insert
Heaps	Delete
Binary Search Trees	Find
AVL Trees	Merge
Hash Tables	Shortest Paths
Graphs	Union
Disjoint Sets	

11

Theory
Graphics
AI
Systems
Applications

Used Everywhere!

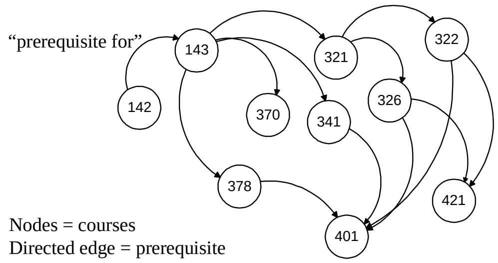
Mastery of this material separates you from:



- ***Perhaps the most important course in your CS curriculum!***
- ***Guaranteed non-obsolescence!***

12

Example 1: Representing Course Prerequisites



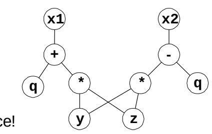
Nodes = courses
Directed edge = prerequisite

13

Example 2: Representing Expressions in Compilers

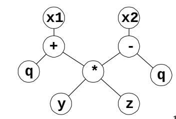
```
x1 = q + y*z
x2 = y*z - q
```

Naive:



y*z calculated twice!

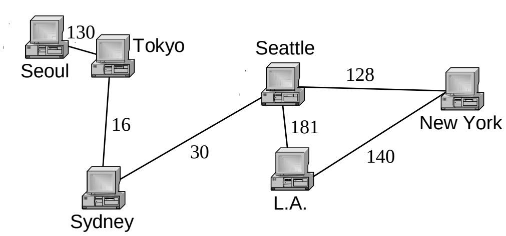
Common Subexpression Eliminated:



Nodes = symbols/operators
Edges = relationships

14

Example 3: Information Transmission in a Network



Nodes = computers
Edges = transmission rates

15

Efficiency: Asymptotic Complexity

Run the program and measure time:

- *Typically insufficient!*
- How big is the sample input? What about other inputs?

Our notion of efficiency:
How does the running time of an algorithm *scale* with the size of its input?

Several ways to further refine:

- worst case
- average case
- amortized over a series of runs

16

Specific Goals of the Course

- Become familiar with some of the fundamental data structures in computer science
- Improve ability to solve problems abstractly
  - data structures are the building blocks
- Improve ability to analyze your algorithms
  - prove correctness
  - gauge (and improve) time complexity
- Become modestly skilled with the UNIX operating system (you’ll need this in upcoming courses)

17

One Preliminary Hurdle

1. Recall what you learned in CSE 321 …
   - proofs by mathematical induction
   - proofs by contradiction
   - formulas for calculating sums and products of series
   - recursion

   Know Sec 1.1 – 1.3 of text by heart!

18

A Second Hurdle

2. Unix

Experience 1975 all over again!

– Try to login on *attu.cs*, edit, and compile your favorite "hello world" program right away

Get help at the UNIX tutorial (tomorrow?)

– Programming Assignment 1 (to be released Wed)

– Bring your questions and frustrations to Section on Thursday!

19

A Third Hurdle: Java

```
Public class Set_of_ints {
  Public void insert( int x );
  Public void remove( int x ); … }
```

Review the syntax (see chapter 1)

Run your first program (assignment 1)

20

Java ≠ Data Structures

One of the all time great books in computer science:

The Art of Computer Programming (1968-1973)

by Donald Knuth

Examples in assembly language (and English)!

Very little about Java in class.

Weiss textbook's code – don't get bogged down!

21

Today's Outline

- Administrative Info
- Survey
- Overview of the Course
- What is an algorithm? ADT? Data structure?
- Stacks and queues

22

What is an Algorithm?

- ???

23

According to …

- According to Mirriam-Webster, an *algorithm* is …
  - a procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation
  - (*broadly*) a step-by-step procedure for solving a problem or accomplishing some end especially by a computer

24

Concepts vs. Mechanisms

- Abstract
- Pseudocode
- Algorithm
  - A sequence of high-level, language independent operations, which may act upon an abstracted view of data.
- Abstract Data Type (ADT)
  - A mathematical description of an object and the set of operations on the object.

- Concrete
- Specific programming language
- Program
  - A sequence of operations in a specific programming language, which may act upon real data in the form of numbers, images, sound, etc.
- Data structure
  - A specific way in which a program's data is represented, which reflects the programmer's design choices/goals.

25

ADT's vs Data Structures

- List ADT
  - Stack ADT
  - Queue ADT
- Collection ADT
  - Stores objects without duplicates
- Dictionary ADT
  - Stores (Key, Value) pairs
  - *Alternatively*: Maps Keys to Values
- Priority Queue ADT
  - Stores objects, and supports efficient removal of objects based upon some kind of ordering
- Graph ADT
- … and even more!

- Linked List
- Circular Array
- Binary Search Tree
- Splay Tree
- Hash Table
- Leftist Heap
- Skew Heap
- Adjacency Matrix

- … and lots more!

So… which ADT's do these data structures implement?

26

Why So Many Data Structures?

Ideal data structure:

"fast", "elegant", memory efficient

Generates tensions:

- time *vs.* space
- performance *vs.* elegance
- generality *vs.* simplicity
- one operation's performance *vs.* another's

The study of data structures is the study of tradeoffs. That's why we have so many of them!

27

ADT Presentation Algorithm

1. Present an ADT
2. Motivate with some applications
3. Repeat until it's time to move on:
   a. analyze its properties
   b. develop a data structure and algorithms for the ADT
      i. efficiency
      ii. correctness
      iii. limitations
      iv. ease of programming
4. Contrast strengths and weaknesses

28

First Example: Queue ADT

- **Queue operations**
  - create
  - destroy
  - enqueue
  - dequeue
  - is_empty

- **Queue property**: if x is enQed before y is enQed, then x will be deQed before y is deQed

<u>FIFO</u>: First In First Out

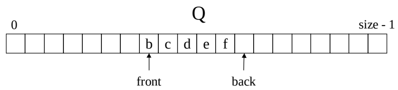
29

Applications of the Q

- Hold jobs for a printer
- Store packets on network routers
- Make waitlists fair
- Breadth first search

30

Circular Array Q Data Structure




```
enqueue(Object x) {
   Q[back] = x ;
   back = (back + 1) % size
  }

dequeue() {
   x = Q[front] ;
   front = (front + 1) % size;
   return x ; }
```

How test for empty list?

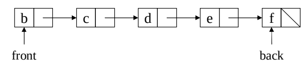
How to find k^{th} element in the queue?

What is complexity of these operations?

Limitations of this structure?

31

Linked List Q Data Structure




```
void enqueue(Object x) {
  if (is_empty())
     front = back = new Node(x)
  else
     back->next = new Node(x)
     back = back->next
}
```

```
Object dequeue() {
  assert(!is_empty)
  return_data = front->data
  temp = front
  front = front->next
  delete temp
  return temp->data
}
bool is_empty() {
  return front == null
}
```

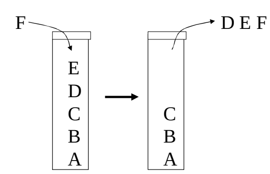
32

Circular Array vs. Linked List

33

Second Example: Stack ADT

- **Stack operations**
  - create
  - destroy
  - push
  - pop
  - top
  - is_empty



- **Stack property**: if x is on the stack before y is pushed, then x will be popped after y is popped

  LIFO: Last In First Out

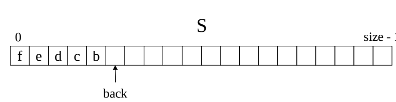
34

Stacks in Practice

- Function call stack
- Converting recursion to iteration
- Balancing symbols (parentheses)
- Evaluating Reverse Polish (postfix) Notation
- Depth first search

35

Array Stack Data Structure

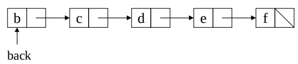



```
void push(Object x) {
   assert(!is_full())
   S[back] = x
   back++
}
Object top() {
   assert(!is_empty())
   return S[back - 1]
}
```

```
Object pop() {
   back--
   return S[back]
}
bool is_empty() {
   return back == 0
}
bool is_full() {
   return back == size
}
```

36

Linked List Stack Data Structure




```
void push(Object x) {
   temp = back
   back = new Node(x)
   back->next = temp
}
Object top() {
   assert(!is_empty())
   return back->data
}
```

```
Object pop() {
   assert(!is_empty())
   return_data = back->data
   temp = back
   back = back->next
   return return_data
}
bool is_empty() {
   return back == null
}
```


Data structures you should already know

- Arrays
- Linked lists
- Queues
- Stacks

To Do

- Return your survey before leaving!
- Check out the web page
- Come to the Unix tutorial - TBD
- Sign up for the cse326 mailing lists
- Log on to the PCs in rooms 002, 006 or 022 and access instructional UNIX server *attu.cs*
  - If you don't have a CSE account, sign up today!
- Read 1.1-1.3, Chapters 2 and 3 in the book
  - Don't worry, it gets better!