CSCI 136
Data Structures & Advanced Programming

Jeannie Albrecht
Lecture 7
Feb 24, 2014

Administrative Details

• Lab 2 due today
  • Any questions?
  • You have to use “tar” to submit your code this time…be careful!
• Lab 3 – no design doc! But you need to do warm-up problems before lab on Wed
• Extra credit on labs

Last Time

• We began talking about how Vectors are implemented in Java

Today’s Outline

• Finish up Vector implementation
• Learn how to “mathematically” analyze the performance of Vectors
• How long do algorithms take to run?
  • The time-space tradeoff
  • Very important concept in computer science!

Implementing Vectors

• Vectors are really just arrays of Objects
• Key difference is that the number of elements can grow and shrink dynamically
• How are they implemented in Java?
  • What instance variables do we need?
  • What methods? (start simple)
• Constructor(s): Vector(), Vector(size), get(index), set(index, Obj), add(Obj), add(index, Obj), remove(index), isEmpty(), size()
• Using parameterized data types

Extending the Array

• How should we extend the array?
• Possible extension methods:
  • Add one to array when capacity is reached
  • Double array when capacity is reached
• Let’s analyze the two techniques
  • Mathematically
  • Experimentally (speed tests)
**ensureCapacity**

- How to implement `ensureCapacity(int minCapacity)`?
  
  // post: the capacity of this vector is at least minCapacity
  public void ensureCapacity(int minCapacity) {
      // First we need to figure out "newlength"
      int newLength = elementData.length; // initial guess
      if (capacityIncrement == 0) {
          // increment of 0 suggests doubling (default)
          if (newLength == 0) {
              newLength = 1;
          }
          while (newLength < minCapacity) {
              newLength *= 2;
          }
      } else {
          // increment != 0 suggests incremental increase
          while (newLength < minCapacity) {
              newLength += capacityIncrement;
          }
      }
      // assertion: newLength >= elementData.length.
      Object newElementData[] = new Object[newLength];
      int i;
      // copy old data to array
      for (i = 0; i < elementCount; i++) {
          newElementData[i] = elementData[i];
      }
      elementData = newElementData;
      // garbage collector will (eventually) pick up old elementData
  }
  // assertion: capacity is at least minCapacity