Lecture 40: Asymptotic Complexity, Searching, and Sorting

Last time:
1. Overloading and overriding revisited
2. Interfaces and class inheritance
3. Interface hierarchies

Today:
1. Project #8 assigned!
2. Course Reviews
3. Asymptotic Complexity and Linear/Binary Search
4. Bubble Sort
Project #8 Assigned!

- Project due 5/10 at 11pm
- Project is open
- Start before now!
  - Read entire assignment from beginning to end before starting to code
  - Check out assignment now from CVS
  - Follow the instructions exactly, as much of grading is automated
Course reviews

- www.courses.umd.edu/online_evaluation/

- IMPORTANT: Please fill this out for your instructor/professor AND for your TA’s (There are two different online evaluations for you to fill out.)
Recap

- Assessing speed of algorithms:
  - Comparing algorithm speed as data sets get larger and larger
  - $n =$ size of data set
  - $T(n) =$ time a particular algorithm takes to “process” $n$ elements of data
- Big-O Notation:
  - If we have a function that is $O(n^2)$, then the function is NO WORSE than $\theta(n^2)$.
  - It might even be BETTER than $\theta(n^2)$, e.g., $\theta(n \log n)$ or $\theta(n)$.
  - If an algorithm is $\theta(n^2)$, we say it is $O(n^2)$, $O(n^3)$, $O(n^4)$, …
- Linear Search: $O(n)$
- Binary Search: $O(\log n)$
Asymptotic Complexity:
Linear Search

- We are assuming Linear Search takes one second at each step, so average running time $T(n) = n/2$
- On different machines, the ACTUAL runtime will be different.
- But we can say that the runtime will ALWAYS be linear, no matter what machine we use.
  - $T(n) = .0001n + .005$
  - $T(n) = .00000007n + .001$
  - What do the graphs look like above?
- The linear search is $O(n)$ — “runs in linear time”
Asymptotic Complexity: Binary Search

- Same idea!
- Different machines run in different times, but it will always be “logarithmic running time”
- Note different log curves are lumped into the same category:
  - Are $T(n) = \log_2 n$ and $T(n) = \log_7 n$ in the same category?
  - Yes: $\theta(\log n)$
- How about $T(n) = 2^n$ vs $T(n) = 3^n$ ??
  - These are in different $\theta$ categories!
Asymptotic Complexity and Competing Machines

- Asymptotic complexity characterizes algorithms, even on competing machines!
- What happens if we compare Atari 800 to Cray Supercomputer for linear and binary search?
- Two cases to compare:
  - Run same algorithm on two different machines:
    - Cray runs linear search
    - Atari runs linear search
    - As \( n \to \infty \) Graphs same shape, but one is clearly better
  - Run different algorithms:
    - Cray runs linear search
    - Atari runs binary search
    - As \( n \to \infty \) Atari wins. (But how long does it take???)
- What if we get an even faster machine than Cray Supercomputer?
  - Slope changes and cross-over point is further out—but eventually there will be a cross-over point!
- Take-home point: If you have two algorithms from two different categories, and you run them on two different computers, eventually (for sufficiently large values of \( n \)), the faster algorithm will win.
  - Algorithm A: \( O(n^{100000}) \)
  - Algorithm B: \( O(1.001^n) \)
  - Plug in \( n = 100 \). For sufficiently large values of \( n \), A will be faster!
What happens in the real world?

- In the real world, if you measure the running time of your algorithm for various values of n, it probably won’t EXACTLY match some known function.
- Graph is “wiggly” at first, but as n grows, approaches some function asymptotically.
- We are concerned with what the graph looks like as \( n \to \infty \).
Bubble Sort

● Fill in the algorithm for Bubble sort:
  
  ```java
  public static void BS (Comparable[] c)
  {
  }
  }
  
  ● What is the run time? (Your homework. 😊 )