CS 209
Data Structures and Mathematical Foundations

02 / 16 / 2024

Instructor: Michael Eckmann

Today's Topics

- Questions/Comments?
- Thoughts about run-time of methods in Linked List
- Run-time analysis by counting the work done
  - Selection sort (we wrote a version of this the 3rd day of class)
  - Insertion sort
  - Linear search
  - Binary search
- Algorithm analysis in general

Linked Lists

- head refers to the first Node in the LinkedList
- Each Node's next refers to the next Node in the LinkedList
- The last-Node-in-the-LinkedList's next has the value None - yesterday's lab had you maintain a link to this last node, called tail.

Linked Lists

- Let's consider "runtime" of the operations (assume number of elements is n)
- Which methods benefitted from the tail reference?

Algorithm Analysis

- An **algorithm** is a specific set of instructions for solving a **problem**.

- Problem vs. algorithm --- a problem is not the same as an algorithm. Example: Sorting is a problem. An algorithm is a specific recipe for solving a problem.
  - bubbleSort, insertionSort, etc. are different algorithms for sorting.
- So, when we're analyzing the running time --- we're analyzing the running time of an algorithm, not a problem.

Algorithm Analysis

- Algorithms are often analyzed for
  - the amount of time they take to run
  - the amount of (extra) space required while running

  - Input size is n. e.g. when an algorithm works on a list, the number of elements of the list is n

Algorithm Analysis

- When we are considering the run-time of an algorithm that works on a list, we say the number of items in the list is **n**
- When we count up the amount of work in an algorithm, we consider certain operations like comparisons, assignments, arithmetic to take some (different) constant amounts of time.

Algorithm Analysis

- We can do **best case** analysis (when the algorithm does the least work / that is, fastest) and **worst case** analysis (when the algorithm does the most work / that is, slowest).
- We must always consider the size of the list to be an arbitrary value n.
- We can't set n to some value when considering best case or worst case. e.g. can't say best case of an algorithm is when the list is empty (because you are setting n to 0 in that case.) Can't say best case is when list is length 1 (here you are setting n=1).
- What we can do is set values of the list to cause a best case or worst case situation. We just can't restrict the length of the list to be some particular value (or even some subset of the positive integers).

Algorithm Analysis

- Consider linear search
- I'll write code for linear search on the board
- For a list of size n
- What situation(s) cause the best case (that is, the least amount of work to do)?
- What situation(s) cause the worst case (that is, the most amount of work to do)?

Algorithm Analysis

- Consider linear search
- For a list of size n
- The worst case happens when the key is not found, or it is found in the last slot
  - Result: linear search makes n compares in the worst case
- The best case happens when the key is found in the first slot
  - Result: linear search makes 1 compare in the best case
  - Notice: even when n is large, best case will still be when key is found in first slot

Algorithm Analysis

- Let's look at a sorting algorithm page and focus on SelectionSort and the InsertionSort
- Also we can look at the code for SelectionSort that we wrote earlier this semester
- http://math.hws.edu/eck/js/sorting/xSortLab.html
- We will examine several runs and see how long they take.

Algorithm Analysis

- The following sum appeared when we tried to figure out how many compares happen in SelectionSort
- Sum of i, where i goes from 1 to n = 1 + 2 + 3 + … + (n-1) + n
- = n * (n + 1) / 2
- We can prove it with induction (maybe next time)
- Examples:
  - 1+2+...+10 = (10*11)/2 = 55
  - 1+2+3+4 = (4*5)/2 = 10
  - 1+2+...+100 = (100*101)/2 = 5050

Algorithm Analysis

- Let's now consider binary search
- For a list of size n
- The best case happens when?
  - Result: binary search makes how many compares in best case?
- The worst case happens when?
  - Result: binary search makes how many compares in the worst case