Lecture 35: Searching & Sorting

CS 51G

Spring 2018

Kim Bruce

Announcements

- Test program 2 now live
    - Design due Tuesday, April 24
        - It will not be returned before program is due!
        - Keep a copy for yourself!
    - Due last day of classes
- Regular lab this Friday
    - But following week's lab devoted to test program

Searching

- Iterative vs Recursive
- Linear vs Binary.
  - Binary requires list be sorted!
- How many comparisons does it take to find an element?
- http://www.cs.pomona.edu/classes/cs051G/demos/SearchSort/search.grace

Timing

Linear search: n comparison in worst case
Binary search: log n comparisons in worst case

search/n	*10*	*100*	*1000*	*1,000,000*
linear(n)	10	100	1000	1,000, 000
Binary (log n)	4	7	10	20

Sorting

- Many kinds
  - Simple sorts: insertion, *selection*
    - take roughly $n^2/2$ comparisons to sort n elements
  - More complex sorts: *merge*, quick sort
    - take roughly n log n comparisons to sort n elements

Selection Sort

- Expressed recursively:
- Find smallest element of list and swap with first element of list.
- Sort the rest of the list in place
- Example:
  - [9,7,3,1,6,4] => [1,7,3,9,6,4] => ... => [1,3,4,6,7,9]
- http://www.cs.pomona.edu/classes/cs051G/demos/SearchSort/sort.grace

Complexity of Selection Sort

- Count number of comparisons in selection sort:
  - $(n-1) + (n-2) + \ldots + 2 + 1 = n(n-1)/2 \approx n^2 / 2$

Insertion Sort

- Alternative simple sort: Insertion sort
  - To sort a list of size n
    - ask assistant to sort last n-1 elements
    - you put the (original) first element where it belongs in list
  - Iteratively:
    - Put first two in order
    - Insert third where belongs in first two
    - Insert fourth where it belongs in first three
    - ...
  - Comparisons: $1 + 2 + 3 + \ldots + (n-1) = n(n-1)/2 \approx n^2 / 2$
  - On average twice as fast as selection sort.

Questions?