Lecture 36: More 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
- Apples lab this Friday
  - Focus on files and strings
- Exercise 19.6.3

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

Last time!

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
    - ...
  - Worst case comparisons: $1 + 2 + 3 + \ldots + (n-1) = n(n-1)/2 \approx n^2 / 2$
  - On average twice as fast as selection sort.

Merge Sort

- Divide list in half,
  - Sort first half
  - Sort second half
  - Merge two sorted halves together
  - See sort demo:
  - http://www.cs.pomona.edu/classes/cs051G/demos/SearchSort/sort.grace

Complexity of Merge Sort

- Merge two lists of total size n takes $\leq$ n-1 compares
- Let T(n) = # comparisons to merge sort list of size n.
- T(0) = T(1) = 0. Why?
- $T(n) \leq T(n/2) + T(n/2) + (n-1)$
- Claim: $T(n) < n \log_2 n$

QuickSort

- Another divide and conquer sort
  - not in sort Grace program
  - Move all small elements to left side of list, all large elements on left.
  - Sort small and then sort large
  - Done!
  - Also takes about n log n compares on average
    - Though worst case is roughly n^2.
    - Happens when list already sorted in either direction

Which sort when?

- Short lists (50 or fewer elements):
  - Selection sort or insertion sort are faster.
  - If partially sorted, insertion can be much faster than selection
- Long lists (50 or more)
  - QuickSort is fastest on average
    - But worst case is worse than selection/insertion
  - Merge sort always roughly n log n, so better if can't afford long delays.
  - Merge sort takes more space (extra list of size n)

Next time: Python

Questions?