Lecture 9: Merge Sort & Correctness

CS 62
Fall 2017
Kim Bruce & Alexandra Papoutsaki

Assignment 3

- On-disk sorting: What to do when more data than can fit in memory of computer?
  - Break into chunks that will fit in memory, sort chunks, and copy into new files: 0.tempfile, 1.tempfile, ...
    - Keep ArrayList of files
  - Merge files together until one big sorted file.
  - Note can't keep file open as both read and write!

Assignment 3

- Read info on File I/O in Java and file systems in appendix to assignment. See on-line Streams cheat sheet!
- Lab 3: More complexity/timing (sorting)

Review: Selection Sort

- Find largest element, put it last, sort the rest!
- More carefully: To sort array[0..n-1]:
  - if n > 0:
    - Swap largest element with array[n-1]
    - Recursively sort array[0..n-2]
- Complexity: $O(n^2)$
- Correctness

Merge Sort

- Example of Divide & Conquer algorithm:
  - Divide array in half
  - Sort each half
  - Merge halves together into completely sorted array
  - See code on line

Correctness

- Course-of-values induction:
  - P(n): If high - low = n then sortHelper(data,low,high) will result in data[low .. high] being correctly sorted
  - For simplicity, assume merge is correct
  - Assume P(k) for all k < n, show P(n).
    - If n = 0 or 1 then (correctly) do nothing. Assume n > 1
    - Call sortHelper(data, low, mid) and sortHelper(data, mid+1, high)
    - where mid = low + (high - low)/2.
    - Hence mid-low < n, high-(mid+1) < n. *Convince yourself!!*
    - By induction data[low..mid] and data[mid+1 .. high] now sorted.
    - call merge(data, low, mid, high) and, by assumption on merge, data[low .. high] now sorted! Thus P(n) true.

Complexity

- Claim mergeSort is O(n log n)
  - where log is base 2
- Intuitive proof
- Careful proof by induction (assuming n is a power of 2, e.g. $n = 2^m$) on board.
- Note: merge of two lists of combined size n takes ≤ n -1 compares.

When we write log n in CS, we mean $\log_2 n$

Complexity

- P(m): if data has 2^m elements then mergesort makes $< m * 2^m$ comparisons of elements.
- Assume P(k) for all $k < 2^m$. Prove P(m)
  - P(0), P(1) clear
  - Show P(m),
  - Sort first half, second half, and then merge
    - size $2^m / 2 = 2^{m-1} < 2^m$, so by induction, each takes $< (m-1) * 2^{m-1}$ comparisons
    - Therefore comparisons $< (m-1) * 2^{m-1} + (m-1) * 2^{m-1} + (2^m - 1)$

Finish Algebra

Compares < $(m-1) * 2^{m-1} + (m-1) * 2^{m-1} + (2^m - 1)$

$= (2m) * 2^{m-1} - 1$

$= m * 2^m - 1$

$< m * 2^m$

Thus P(m) true. Note if $n = 2^m$ then merge sort takes (log n) * n comparisons because m = log n. Easier to write as n * (log n)

Binary Search

- If time:
  - Search for element in sorted list
  - Linear search is O(?)
  - Binary search is O(?)