Agenda:

- Merge sort (analysis later)
 - a quick review
 - recursion
 - correctness
- Asymptotic notations

Reading:

• Textbook pages 28 - 61

```
Merge sort pseudocode
```

Lecture 4: Merge Sort & Asymptotic Notations Algorithm analysis issues (review):

- Issues:
 - correctness
 - resources used (time & space)
 - optimality
- Estimating resources used
 - input size n: time T(n) & space S(n)
 - worst/best/average case (WC/BC/AC)
 - machine independent computational model
- Model of computation
 - simple (architecture, instruction set)
 - reflective (typical machine, accurate estimates)
 - our choice RAM (random access machine)
 - two versions: uniform cost & log cost
 - * choose version depending on applications
 - * astronomical numbers log cost RAM
 - * reasonable size numbers uniform cost RAM

Algorithm running time analysis

- what have been counted?

E.g.,

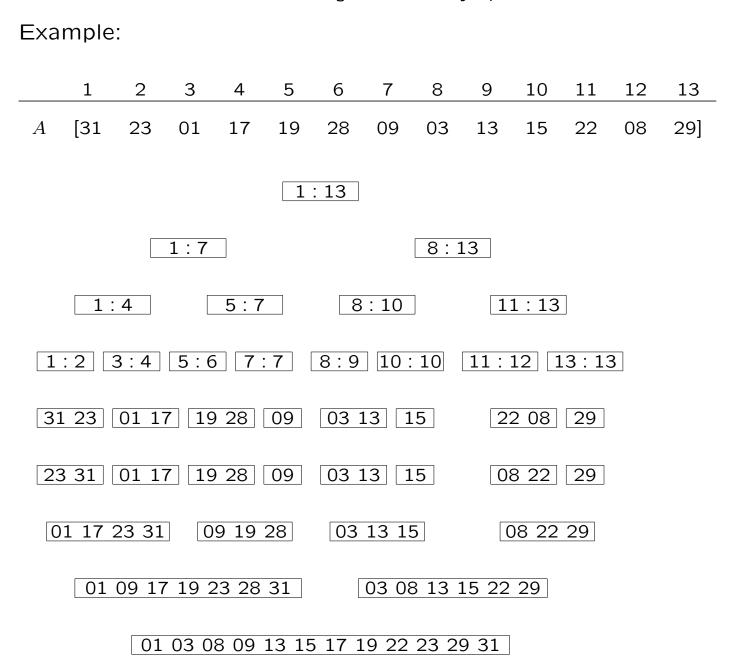
- RAM instructions?
- log RAM instructions?
- Data moves?
- Data comparisons?
- Arithmetic operations?
 - additions? subtractions?
 - multiplications? divisions?
- Pentium IV clock cycles? etc.

Merge sort, the big idea — divide-and-conquer:

- Divide the whole list into 2 sublists of equal size;
- Recursively merge sort the 2 sublists;
- Combine the 2 sorted sublists into a sorted list.

Make sure you do know how to combine !

- This is the idea of *recursion*.
 - a programming technique (not really a design technique)
 - recursion trees
 - recurrence relations



Lecture 4: Merge Sort & Asymptotic Notations

5

Lecture 4: Merge Sort & Asymptotic Notations Asymptotic notations, motivations:

- Analysis of algorithms becomes analysis of functions:
 - e.g., f(n) denotes the WC running time of insertion sort g(n) denotes the WC running time of merge sort
 - $f(n) = c_1 n^2 + c_2 n + c_3$ $g(n) = c_4 n \log n$
 - Which algorithm is preferred (runs faster)?
- To simplify algorithm analysis, want function notation which indicates *rate of growth* (a.k.a., *order* of complexity)

$$O(f(n))$$
 — read as "big O of $f(n)$ "

- roughly, The set of functions which, as n gets large, grow no faster than a constant times f(n).
- precisely, (or mathematically) The set of functions $\{h(n) : N \to R\}$ such that for each h(n), there are constants $c_0 \in R^+$ and $n_0 \in N$ such that $h(n) \leq c_0 f(n)$ for all $n > n_0$.

examples:
$$h(n) = 3n^3 + 10n + 1000 \log n \in O(n^3)$$

 $h(n) = 3n^3 + 10n + 1000 \log n \in O(n^4)$
 $h(n) = \begin{cases} 5^n, & n \le 10^{120} \\ n^2, & n > 10^{120} \end{cases} \in O(n^2)$

Have you understood the lecture contents?

well	ok	not-at-all	topic
			alg analysis in general
			divide-and-conquer: merge sort idea
			merge sort algorithm
			why asymptotic notations
			what $O(f(n))$ means