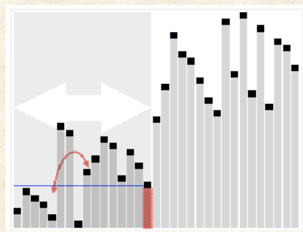
UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 202: Introduction to Computation

Professor Andrea Arpaci-Dusseau

How can computation… sort data faster for you?



Previous Lecture

Two intuitive, but slow sorting algorithms

Selection sort:

- Repeat for each key in list
  - Find minimum key in unsorted portion
  - Move to next position of sorted portion

Insertion sort:

- Repeat for each key in unsorted list
  - Insert into its correct position in sorted portion

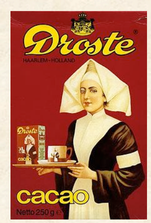
Both algorithms $O(N^2)$ where N is length of list

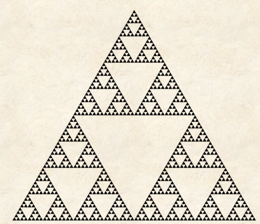
Sorting Algorithms: Speed Comparison

Recursive Algorithms

Algorithm is recursive if can be defined by:

- Simple base case
- Set of rules reducing other cases toward base case





Recursion: If you still don't get it, see: "Recursion".

Recursive Definition of Factorial

Example: Fact(5) = 5! = 5 * 4 * 3 * 2 * 1

Recursive definition:

- Fact(1) = 1 [base case]
- For all integers n > 1: Fact(n) = n * Fact (n-1)

Fact(5) = ??

= 5 * Fact (4)

= 5 * 4 * Fact(3)

= 5 * 4 * 3 * Fact(2)

= 5 * 4 * 3 * 2 * Fact (1)

= 5 * 4 * 3 * 2 * 1 Recursion ends!

Merge Sort Algorithm: Uses Recursion

Base case:

- If list of length 0 or 1, done (sorted)

Otherwise:

- Divide unsorted list of size M into two sublists of size M/2
- Sort each sublist recursively using mergesort
- Merge two sublists back into one sorted list

How to merge two lists into one?

Merging Two Sorted Runs

→ 2	→ 4
5	8
6	9
10	13
End	End

Algorithm: Compare 1st element of each list, remove the smaller as next element of sorted run

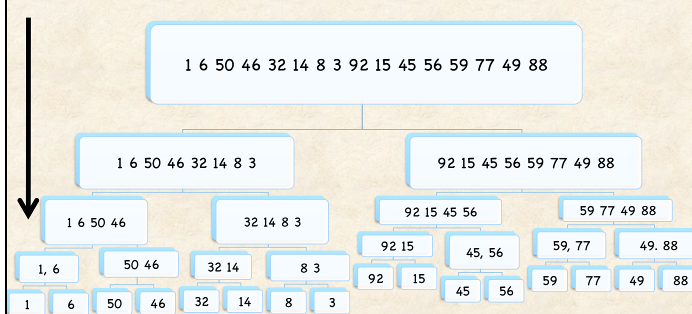
Very efficient! Very few comparisons needed for merge

How many comparisons needed to create list of size N?

O(N) comparisons

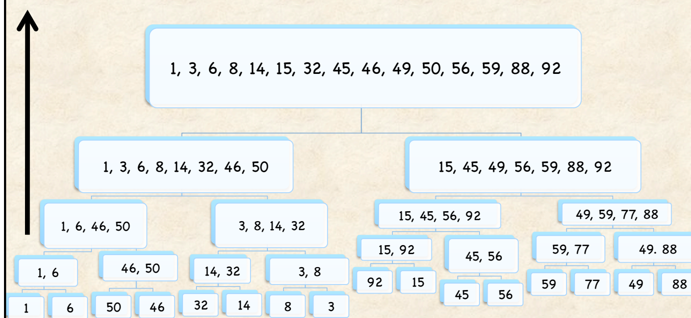
Merge Sort: Recursively Divide Down Tree

Sort keys: 1 6 50 46 32 14 8 3 92 15 45 56 59 77 49 88

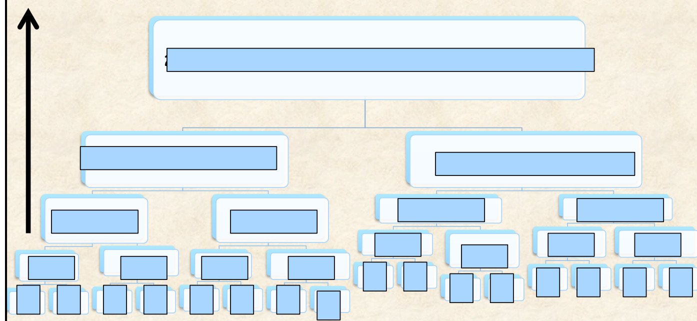


Merge Sort: Merge Two Lists Up Tree

Sort keys: 1 6 50 46 32 14 8 3 92 15 45 56 59 77 49 88

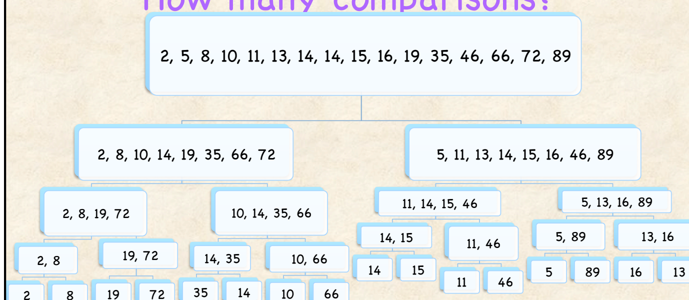


Merge Sort: Example



Sort keys: 2 8 19 72 35 14 10 66 14 15 11 46 5 89 16 13

Merge Sort: How many comparisons?

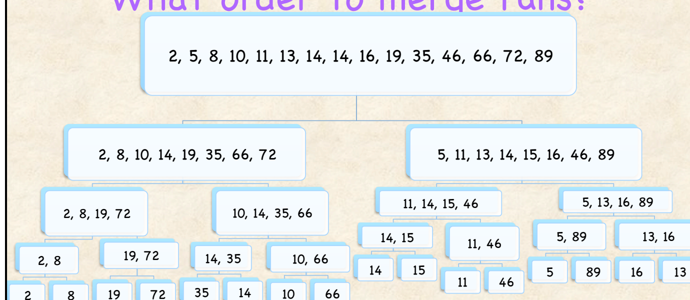


How high (or deep) is the tree?
- $\log_2 N$

How many comparisons to create next level? (last run? 2nd-to-last two runs?)
- last run: N, 2nd to last two runs: 2 * N/2, next: 4 * N/4 … always N!

Total comparisons?
- $N \log_2 N$

Merge Sort: What order to merge runs?



Which runs can be merged independently of others?
- All runs at same level are independent!
- Must create runs lower in the tree first!

Why is this a good property?
- Can merge runs in parallel!
- Great for multiprocessors, multi-cores, clusters of machines

Algorithm Comparison

Quicksort (Qsort) Algorithm: Recursive

Base case: list of size one is sorted by definition

Otherwise:

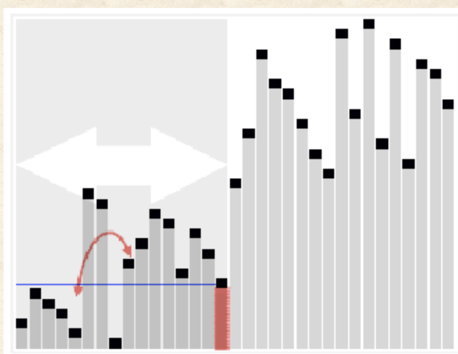
Pick an element (pivot) from list

Reorder:

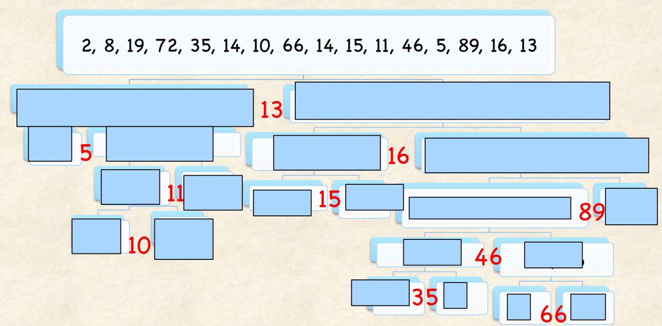
- All keys < pivot → move key before pivot
- All keys > pivot → move key after pivot
  - Equal values can go either way
- Pivot is now in its final sorted position

Recursively sort (w/ quick sort!) two sub-lists

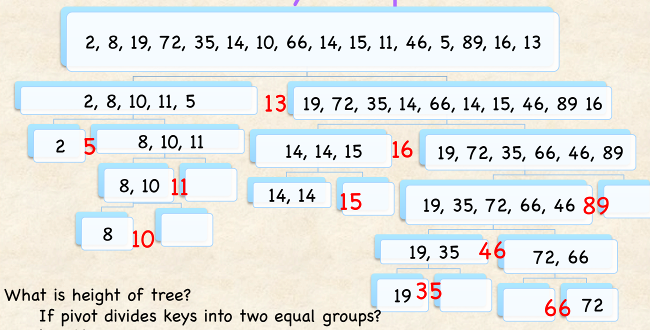
Quicksort Demo



Quicksort Example



Quicksort: How many comparisons?

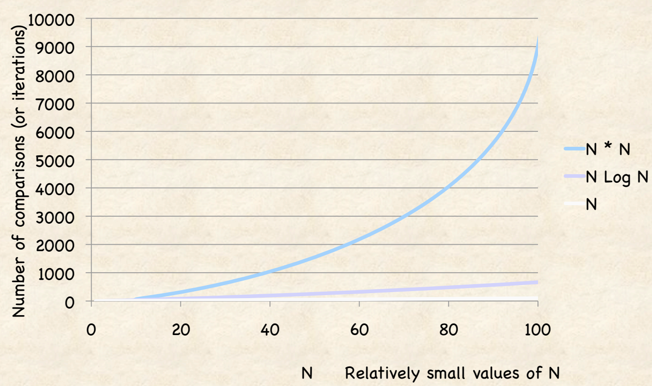


What is height of tree?
If pivot divides keys into two equal groups?
$\log_2 N$
How many comparisons to form new level of tree?
N
Total comparisons?
$N \log_2 N$

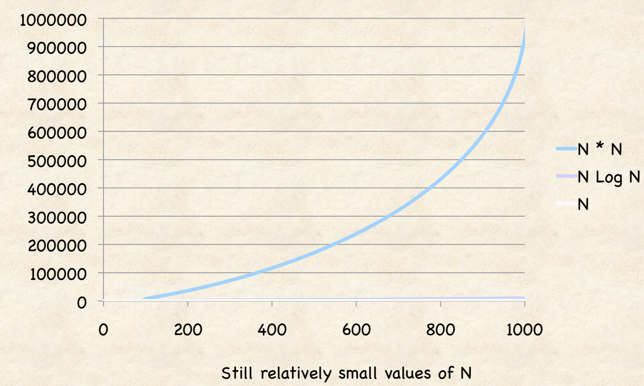
Sorting Algorithm Comparison

	Selection Sort	Insertion Sort	Merge Sort	Quick Sort
Worst case?	$O(N^2)$	$O(N^2)$	O (N log N)	$O(N^2)$ If pick bad pivot
Best case?	$O(N^2)$	O(N) If sorted already	O (N log N)	O (N log N)
Average case?	$O(N^2)$	$O(N^2)$	O (N log N)	O (N log N)

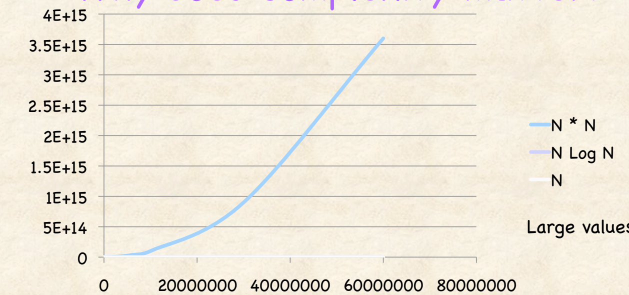
Why does Complexity matter?



Why does Complexity matter?



Why does Complexity matter?



Large values of N

How long to sort 60,000,000 keys?
 Assume 3 billion (3 * 10^9) comparisons per second
With N* N algorithm: Approx 3 * 10^{15} comparisons
 Requires 10^6 seconds = 280 hours! More than 10 days!
With N Log N algorithm: Approx 1.5 * 10^9 comparisons
 Requires 0.5 seconds!
N Log N grows very slowly with N… Practical for large N

NOW-Sort: World Record Holder

Sorted 1 million keys (1997)
- Disk-to-disk
- < 2.5 seconds
- 100 machines on network

Merge sort works well here
- Each machine starts with 1/100 of keys (and data!) on local disk
- Sorts its own keys
- Each sends sorted run of keys (and data!) to destination machine
- After receive all keys, each machine:
  - Merge 100 sorted runs

Check-Up




A. How many times will blocks in largest innermost loop execute?
   - That is, what is the value of counter when script terminates?

C. What is the order of magnitude for the number of operations in each as a function of N?
   - Remember: When calculating order of magnitude, disregard constant factors.
   - Hint: If runs in time independent of N, then O(1)

Announcements

Sorting algorithms
- O(N^2) sorting algorithms
  - Selection sort: Find minimum and make next
  - Insertion sort: Take next and insert in correct place
- O(N log N) sorting algorithms (expected, not worst-case)
  - Merge sort: Recursively combine sub-lists into larger lists
  - Quicksort: Recursively partition list into sub-lists around pivot

Announcements
- Homework 5: Congrats to Fong Lor, Jake Hilborn, and Kameko Blair
- Homework 6: No Extra Credit
  - email me if you think your project showed creativity, may re-evaluate
- Homework 7: Due Friday – No Programming
  - Explore Google Trends, Understand basic sorting algorithms, Reflect on Technology and Education (make sure you can watch Friday video)