CS 61B Spring 2019

More Sorting

Exam Prep 13: April 22, 2019

1 Quickselect (Summer 2016, Final)

The Quicksort algorithm can be modified for finding the k-th smallest element in an array. This is called the Quickselect algorithm, which finds the item at sorted index $j = k - 1$. The partition step is the same as Quicksort, and only differs in the recursive call (recursing on the partition that contains the k-th smallest element). Assume the partitioning is done using the three-way partitioning strategy.

(a) Using the first element of the list as the pivot, show how the list is partitioned at each step when Quickselect ($j = 5$) is called, trying to find the sixth smallest element. Circle the partition that is recursed on at each step.

42 93 50 39 81 94 23 28 95 89

Left Partition	Pivot Partition	Right Partition	j
39 23 28	42	93 50 81 94 95 89	5
50 81 89	93	94 95	1
	50	81 89	1
	81	89	0

(b) What is the best and worst case runtime of Quickselect?

Best: $\Theta(N)$
$Worst: \Theta(N^2)$

(c) Assume we use Quickselect to find the median of an array and always choose the median as the pivot. In this case, what would be the best and worst case runtime of Quicksort?

Best: $\Theta(N \log N)$
$Worst: \Theta(N^2)$

2 Radix Sorts (Summer 2018, Final)

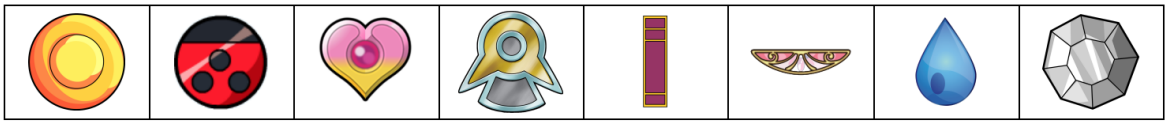
Suppose we're considering alternatives to the counting sort algorithm used in LSD radix sort. When used as the sorting algorithm in LSD radix sort, which of the following sorts is guaranteed to correctly sort a list in $\Theta(WN \log N)$ time? Assume W is the length of the longest key, N is the number of keys, and the radix, R, is constant.

Mergesort

3 Insertion Sort (Summer 2017, Final)

Consider the following unsorted array, and the same array after 4 iterations of insertion sort. The first iteration starts at index 1 (the second element in the array). Assume no two elements are equal.

Unsorted:



After 4 iterations of insertion sort

For each row, fill in the bubble that corresponds to the relationship between the symbols. If there is not enough information, fill in the ? bubble.

Symbol 1	<	>	?	Symbol 2
	○	○	●	
	○	●	○	
	●	○	○	

○ True / ● False:  must be the smallest element in the array.

4 Sorting (Spring 2017, Final)

(a) Suppose we have N items we want to sort, where N is very large. For each scenario below, write the number of the "best" sort to sort the numbers or objects, and give the running time for the absolute worst case as a function of N. There may be multiple correct answers, and the correct answer may even be ambiguous. Running time may not be the only consideration for best. In all cases, assume were using Java.

Choose from among 1: Insertion sort, 2: Merge sort, 3: Quicksort (with Hoare partitioning), and 4: LSD radix sort. Assume that we want stability when potentially useful.

Note: A `BigInteger` is an "immutable arbitrary precision integer". It can represent any integer, not just those that fit into 32 bits.

Scenario (i.e. What You're Sorting)	Best Sort	Running Time
Array of N integers whose max value k is a constant. Do not include k in your runtime.	4	O(N)
Array of N `BigIntegers`[1] whose max value is N^3. Assume comparison takes `log(N)` time.	4	O(N Log N)
Array of N objects that implement `Comparable`, assuming comparison takes constant time.	2	O(N Log N)
Doubly linked list of N objects that implement `Comparable`, assuming constant time comparison and all variables `public`.	2	O(N Log N)
Array of 10 `Strings` of length W. Give runtime in terms of W.	1	O(W)
Array of N objects that implement `Comparable` with $\Theta(\sqrt{N})$ inversions, assuming compare takes constant time.	1	O(N)
Array of N objects that implement `Comparable` with $\Theta(N^2)$ inversions, assuming compare takes constant time.	2	O(N Log N)

(b) A sort is **monotonically improving** if the number of inversions never increases as the sort is executed. Which sorts from the list below are monotonically improving? Assume that all sorts are as presented during lecture on arrays. Assume insertion sort and selection sort are in-place. Assume heapsort is in-place and that the array acts as a max heap. Assume that Quicksort is non-randomized, uses the leftmost item as pivot, and uses the Hoare partitioning strategy.

NOTE: Solutions should also include quicksort.

■ Insertion sort ■ Selection sort ☐ Heapsort ☐ Quicksort ☐ LSD Sort ■ MSD Sort