QuickCheck 08: Solutions

Due: 8:00 am on Thursday, Mar 5, 2020

QuickChecks must be scanned and submitted online via Gradescope. If you have a smartphone, you can follow these steps to scan using an app: https://www.gradescope.com/help#help-center-item-student-scanning. Otherwise, there are scanners located at various libraries on campus which can be found here: https://finance.uw.edu/c2/printing-copying/dawg-prints-copy-locations. Make sure that the gray border around the edge of this page is visible in your scanned document.

1. Worst Case

Give the worst possible order of input for insertion sort with the following integers: 1, 2, 3, 4, 5, 6, 7. Assume that the result of sorting should be in ascending order.

Solution:

```
[7,6,5,4,3,2,1]
```

2. Sorting Decisions

For each scenario below, fill in all squares with the letter corresponding to the sorting algorithm that would perform the best. Some of the questions may have multiple answers, in which case you should fill in multiple squares.

A: Insertion Sort, B: In-place Heapsort, C: Merge Sort, D: Selection Sort, E: Naive Quicksort, F: None of the Above

(a) Java integer array with a length of 5. A ☐ B ☐ C ☐ D ☐ E ☐ F ☐

(b) An array of comparable UW classes (objects), that are already sorted by number and need to be additionally sorted by department. For example, the input `[CSE 142, CHEM 142, CSE 143, CSE 373]` would result in the output `[CHEM 142, CSE 142, CSE 143, CSE 373]`.

A ☐ B ☐ C ☐ D ☐ E ☐ F ☐

(c) We need to sort integers in $\Theta(\log N)$ time for the best case. A ☐ B ☐ C ☐ D ☐ E ☐ F ☐

(d) Sorting doubles with a $\Omega(N \log N)$ best-case lower bound. A ☐ B ☐ C ☐ D ☐ E ☐ F ☐

Solution:

(a) A. Insertion sort is fast for low N values

(b) A, C, E. Because the values are not primitives, we need a stable sort, so insertion sort, merge sort, and naive quicksort do the job!

(c) F. None of the above sorts can sort in $\Theta(\log N)$ time

(d) C, D, E. Merge sort, selection sort, and naive quicksort

3. Heapsort

List out the steps of sorting the array `[5, 0, 1, 3]` into ascending order using heap sort with a **max heap**.

The first step should be the array after the intial `buildHeap`, and each successive step should be the array after `removeMax`. You may not need every line provided to write your solution.

(Tip: Draw out the heap. Make sure your first step is `buildHeap`!)

`Step 1:`					(after `buildHeap`)
`Step 2:`					(after `removeMax`)
`Step 3:`					(after `removeMax`)
`Step 4:`					(after `removeMax`)
`Step 5:`					(after `removeMax`)
`Step 6:`					(after `removeMax`)

Solution:

5 - 1 - 3 - 0 : `buildHeap` turns the array into a valid heap
3 - 1 - 0 - 5 : `removeMax(5)`, then percolate 0 down
1 - 0 - 3 - 5 : `removeMax(3)`, then percolate 0 down
0 - 1 - 3 - 5 : `removeMax(1)`
0 - 1 - 3 - 5 : `removeMax(0)`