Stacy Nguyen

Design Patterns & Data Structures

Spring 2020

**Homework 8 - Sorting**

All of the entries below are measured in seconds, and I ran 100 runs to find the average each time.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Presorted** | **Random (avg of 100 runs)** | **Reverse order** |
| Insertion Sort | 0.00041528001000000014 | 0.009113723790000002 | 0.017088847000000004 |
| Selection Sort | 0.12372612263 | 0.012893584089999998 | 0.012351424180000006 |
| Bubble Sort | 0.00014580431 | 0.053786605730000006 | 0.09080518341999999 |

**Written HW**

1) Which of the following tasks would be faster on sorted data?

a. Finding the smallest element

b. Calculating the average of a set of values

c. Finding the median value

d. Checking for the existence of a particular element

e. Finding the most common element

My answer: a, c, d, e

2) What sorting algorithm may be best for the following situations?

a. You have 100 computers to split up the sorting on

b. You have a set of small integers, unique 1-1000

c. You have a set of floats/doubles from 1-1000

d. You have a nearly sorted list

A. Merge sort

B. Quick sort

C. Radix sort

D. Insertion sort

3) Which of the algorithms would be easily adaptable to a doubly linked list?

Merge sort

4) Come up with an example of a 10 element array that is the best case scenario for a

bubble sort, and one that’s a worst case scenario for a bubble sort.

Best case: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Worst case: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]