PROBLEM SET 3
Due: Friday, October 4

Important: On Friday, October 4, you will receive a take-home exam. In order to give you maximal time to work on your exam, I require Problem Set 3 to be turned in no later than 6pm on Friday, October 4. At that time, solutions to Problem Set 3 will be made available.

Reading: Chapters 7 & 8

Animations: There are a number of animations in the Animations subfolder of the CS231 folder that are helpful for learning this material. Check out: Asymptotic Notation, Heap Sort, Insertion Sort, Merge Sort, Quick Sort, and Sorting Analysis.

Suggested Problems: 7.1-5; 7.2-1; 7.3-1; 7.4-1 7.4-2; 7.5-3; 7.5-6; 8-1, 8-2.

Problem 1 [50]: Sorting Analysis

Handout 11 describes six comparison-based algorithms for sorting an array A[1..n] (insertion sort, selection sort, bubble sort, merge sort, heap sort, quick sort). For each of these six algorithms, answer the following questions:

a. What is the worst-case running time of the algorithm? Justify your answer. As part of your answer, describe the structure of an input array on which the algorithm exhibits its worst-case running time.

b. What is the best-case running time of the algorithm? Justify your answer. As part of your answer, describe the structure of an input array on which the algorithm exhibits its best-case running time.

c. What is the average-case running time of the algorithm? Explain.

d. Is the algorithm in-place?

e. Is the algorithm stable?

Problem 2 [20]: Stooge Sort

Do Problem 8-3 on p. 169 of CLR. For part a, give a proof of correctness by induction on the number of elements n = (j - i + 1) in the array. Recall the structure of an induction proof for the correctness of an algorithm:

• Prove that the algorithm is correct on the base case(s).

• Assuming that the algorithm is correct on all inputs whose size is < n, prove that the algorithm is correct on all inputs of size n.

The correctness proof is somewhat subtle, so please explain it carefully. For simplicity, assume that all elements in the array are distinct.
Hint: For each element in the original segment \( A[i..j] \), define its fate as \( lo \) if it would end up in the first segment \( A[i..i+k-1] \) in the final sorted segment, \( mid \) if it would end up in the second segment \( A[i+k..j-k] \) in the final sorted segment, and \( hi \) if it would end up in the third segment \( A[j-k+1..j] \) in the final sorted segment. Use these fate labels to guide your correctness proof.

Grading breakdown on parts of this problem:
- Part a: [10]
- Part b: [7]
- Part c: [3]

**Problem 3 [30]: The Dutch National Flag Problem**

(The following problem was proposed by W.H.J. Feijen and made famous by Edsger W. Dijkstra, both of Dutch origin.)

You are given a row of \( n \) buckets, each containing one ball, which may be either red, white, or blue. Your goal is to rearrange the balls in the buckets such that they appear in the order of the colors on the Dutch national flag: red balls should be grouped on the left, white balls in the middle, and blue balls on the right. The only operation you may use to move balls is \( \text{swap}(i, j) \), which swaps the contents of the \( i \)th and \( j \)th buckets.

a [15]. Give pseudocode for a linear-time algorithm for sorting an array \( B[1..n] \) of balls in the Dutch national flag order. Your algorithm should use only constant space in addition to the given array. *Hint:* study the Lomuto partitioning technique.

b [10]. Prove that your algorithm is correct. *Hint:* use a loop invariant!

c [5]. According to the decision-tree analysis performed in class (and in CLR Section 9.1), the best worst-case running time of a comparison-based sorting algorithm is \( \Omega(n \log(n)) \). Yet the Dutch national flag problem is a linear-time sorting algorithm. Explain how this can be.

**Extra Credit Problem [10]: A Horse of a Different Color**

Consider the following "proof" that all horses have the same color:

Let \( S \) be a set of horses, and let \( P[S] \) be true if all horses in \( S \) have the same color.

(Base Cases)
If \( S \) contains zero or one elements, then \( P[S] \) is trivially true.

(Inductive Case)
Suppose \( P[S_i] \) is true for all sets \( S_i \) containing \( i \) elements, where \( 0 \leq i \leq n-1 \). Given an \( n \)-element set of horses \( S_n \), pick a horse \( h \) from \( S_n \), and pick any two distinct subsets \( A \) and \( B \) of \( S_n \) that (1) have \( n-1 \) elements and (2) contain \( h \). By the inductive hypothesis, \( P[A] \) and \( P[B] \) are true; and since both \( A \) and \( B \) contain \( h \), the color of the horses in \( A \cup B = S_n \) must all be the same. So \( P[S_n] \) is true.

Find and describe the bug in this proof.
Name:

Date & Time Submitted *(only if late)*:

Collaborators *(anyone you collaborated with in the process of doing the problem set)*:

*In the Time column, please estimate the time you spent on the parts of this problem set. Please try to be as accurate as possible; this information will help me to design future problem sets. I will fill out the Score column when grading your problem set.*

<table>
<thead>
<tr>
<th>Part</th>
<th>Time</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Reading</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem 1 [50]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem 2 [20]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem 3 [30]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extra Credit [10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>