

PROBLEM SET 3
Due: Thursday, February 22

Reading: Comparison-Based Sorting notes (Handout #9); CLR chapters 8 & 13

Suggested Problems: 8-1, 8-2, 13.1-3, 13.1-4, 13.1-5; 13.2-1; 13.3-1, 13.3-2, 13-3.5; 13.4-2; 13-2

Problem 1 [40]: Sorting Analysis

Handout 9 describes six comparison-based algorithms for sorting an array $A[1..n]$ (insertion sort, selection sort, bubble sort, merge sort, quick sort, and tree sort). For each of these six algorithms, answer the following questions. In parts a, b, and c, measure running time by the number of comparisons performed.

- a. What is the worst-case running time of the algorithm? *Briefly* 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? *Briefly* 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. (For bubble sort, you need to know the following fact: according to Knuth, at least $k = \frac{1}{2}(n+1)$ passes are required in the average case.)
- d. Is the algorithm in-place? Recall that a sorting algorithm is in-place if it uses only a constant amount of space in addition to the array itself. Note that (non-tail) recursion requires stack space proportional to the height of the invocation tree of the recursive function.
- e. Is the algorithm stable? Recall that a sorting algorithm is stable if it preserves the relative order of elements with the same key.

Problem 2 [27]: 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.

As part of your proof, you should carefully show that the arrays $A[i..j-k]$ and $A[i+k..j]$ mentioned in lines 6--8 have lengths less than n . You should also carefully show the bounds (as a function of n) on the sizes of the three array segments $A[i..i+k-1]$, $A[i+k..j-k]$, and $A[j-k+1..j]$.

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: **[15]**
- Part b: **[10]**
- Part c: **[2]**

Problem 3 [33]: 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 for quicksort on Handout #9.

b [15]. Prove that your algorithm is correct by the method of loop invariants. Recall that this requires four steps:

1. showing that the loop invariants hold the first time the loop is entered.
2. showing that if the loop invariants are true at the beginning of an iteration, they are true after the body of the loop has executed (regardless of the path taken through the body).
3. showing that the algorithm terminates; and
4. showing that the terminating state corresponds to the goal state.

c [3]. 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 $\Theta(n \lg n)$. 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 $A \cap B = S_n$ must all be the same. So $P[S_n]$ is true.

Find and describe the bug in this proof.

Problem Set Header Page
Please make this the first page of your hardcopy submission.

CS231 Problem Set 3
Due Thursday, February 22, 2001

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.*

Part	Time	Score
General Reading		
Problem 1 [40]		
Problem 2 [27]		
Problem 3 [33]		
Extra Credit [10]		
Total		