

# CS381 Homework 2

Out: Sep. 14, Thursday  
Due: Sep. 28, Thursday, in (or before) class  
Submissions will not be accepted afterwards.

## Instructions:

**Read the course policy (including academic dishonesty) in the course webpage at <http://www.cs.purdue.edu/homes/gopal/cs381>.** Text refers to the Introduction to Algorithms (second edition) book.  
**Justify your answers. Show appropriate work.**

**Reading:** Chapters, 2.3, 7, and 9 of Text.

## Problem 1

(a) Prove the correctness of Random-Select algorithm using Induction.

**Input:** A set  $S$  of  $n$  distinct elements, and an integer  $1 \leq i \leq n$ .

**Output:** The  $i$ -th smallest element in  $S$ .

**Random-Select**( $S, i$ )      ( $1 \leq i \leq |S|$ ).

1. If  $|S| = 1$  then return  $S$ .
2. Choose a random element  $y$  uniformly from  $S$ .
3. Compare all elements of  $S$  to  $y$ . Let

$$S_1 = \{x \in S \mid x < y\}, \quad S_2 = \{x \in S \mid x > y\}.$$

4. If  $|S_1| = i - 1$  then return  $y$   
    elseIf  $|S_1| \geq i$  then return Random-Select( $S_1, i$ )  
    else return Random-Select( $S_2, i - |S_1| - 1$ );

(Hint: In class, recall that we argued the correctness of Mergesort and Quicksort (which are also recursive algorithms) using induction.)

(b) What is the worst-case time of the above algorithm? For what input will that occur and why? What is the best-case time of the above algorithm? For what input will that occur and why?

(c) What is the average running time of the algorithm? Show the analysis. (Hint: This was done in class. You have to set up the recurrence for the average running time and solve it.)

## Problem 2

Show that the best-case running time of Quicksort is  $\Omega(n \log n)$ . (Hint: You should show this formally by setting up a recurrence and using induction. In class we showed the worst-case running time of Quicksort is  $O(n^2)$  similarly.)

## Problem 3

Consider the problem of finding the largest and second largest elements from a set of  $n$  elements. Give a divide and conquer algorithm and show that it takes at most  $\frac{3n}{2} - 2$  comparisons when  $n$  is a power of 2.

## Problem 4

Given two SORTED (in ascending order) sets  $A$  and  $B$ , each containing  $n$  numbers, give an  $O(\log n)$ -time algorithm to find the  $n$ th smallest number of all  $2n$  elements in the sets  $A$  and  $B$ . If you want, you can assume that  $n$  is a power of 2. (Hint: Use the fact that the two sets are already sorted. The desired running of  $O(\log n)$  suggests that you have to “prune” a constant fraction (say, nearly  $1/2$ ) of the elements in each step.)

## Problem 5

Suppose you are working for a company that enforces piracy laws. You are given a set of  $n$  CDs and your task is to find whether at least  $n/2$  of the set are the same (this is considered a potential piracy). The only tool you have is a machine that can answer the following type of query: given two

CDs, are they the same or not? Your goal is to design a divide and conquer algorithm that takes  $O(n \log n)$  queries. (Hint: (1) To have at least  $n/2$  identical CDs in the set, at least one of the halves of the set should contain at least  $n/4$  of them. (2) When you solve your subproblems you may want to return more than just saying that are “enough CDs of one kind”.)