# CS264: Homework #10

Due by midnight on Thursday, March 23, 2017

**Instructions:**

(1) Form a group of 1-3 students. You should turn in only one write-up for your entire group. See the course site for submission instructions.

(2) Please type your solutions if possible and feel free to use the LaTeX template provided on the course home page.

(3) All students should complete all of the exercises. Students taking the course for a letter grade should also complete the problems.

(4) Write convincingly but not excessively. Exercise solutions rarely need to be more than 1-2 paragraphs. Problem solutions rarely need to be more than a half-page (per part), and can often be shorter.

(5) You may refer to your course notes, and to the textbooks and research papers listed on the course Web page *only*. You cannot refer to textbooks, handouts, or research papers that are not listed on the course home page. (Exception: feel free to use your undergraduate algorithms textbook.) Cite any sources that you use, and make sure that all your words are your own.

(6) If you discuss solution approaches with anyone outside of your team, you must list their names on the front page of your write-up.

(7) Exercises are worth 5 points each. Problem parts are labeled with point values.

(8) No late assignments will be accepted.

## Lecture 19 Exercises

### Exercise 49

Prove that every distribution $p$ with support size $s$ — i.e., there are only $s$ points $x \in X$ with $p_x > 0$ — has entropy at most $\log_2 s$.

[Hint: prove and use that the entropy function $\sum_{x \in X} p_x \log_2 \frac{1}{p_x}$ is convex.]

### Exercise 50

This exercise considers a minor variant of the optimal search tree problem mentioned in lecture. Recall that a binary search tree on a totally ordered set $X$ is a binary tree with nodes in correspondence with $X$, with the property that every node in the left (right) subtree of a node $x$ must be less than (greater than) $x$. Recall that there are many different binary search trees on the same set $X$ (from long chains to balanced trees). The search time $s_T(x)$ for a node $x$ in a search tree $T$ is one plus its depth in the tree (1 for the root, 2 for the root's immediate children, and so on.). Given a positive probability $p_x$ for every $x \in X$, the *optimal search tree* is the search tree $T$ that minimizes the expected search time for a node $x \in X$, $\sum_{x \in X} p_x s_T(x)$.

Give a dynamic programming algorithm that, given $p_x$'s for all $x \in X$, computes an optimal search tree.

[Hints: shoot for a dynamic programming algorithm that solves $O(n^2)$ subproblems in time $O(n)$ each. (Here $n = |X|$.) To get started, if you happened to know the root node of the optimal search tree, what could you say about its subtrees?]

### Exercise 51

The point of this exercise is to explain why, when constructing a near-optimal search tree, it is enough to consider only the elements with large (at least $1/|X|^\epsilon$) probability, handling the rest via binary search. This justifies the implementation of the approximate search tree construction outlined at the end of lecture.

The precise exercise is the following. Consider a totally ordered set $X$ with $n$ elements. Let $D = \{p_x\}_{x \in X}$ be a probability distribution on $X$ and $S \subseteq X$ the elements with $p_x \geq n^{-\epsilon}$, where $\epsilon > 0$ is an arbitrary constant. Prove that

$$\sum_{x \in S} p_x \log_2 \frac{1}{p_x} + \sum_{x \notin S} p_x \log_2 n = O(H(D)),$$

where the constant hidden in the big-oh notation can depend on $\epsilon$. Explain the relevance of this statement to the algorithm described in lecture.

### Exercise 52

Name at least two places in the proof of this lecture's main result where we used the assumption that the $x_i$'s are independent.

## Lecture 20 Exercises

### Exercise 53

Let $F$ be a distribution on $[0, 1]$ with expectation $\mu$ and $X_1, \ldots, X_n$ be i.i.d. samples from $F$. Prove that if $n \geq \frac{c}{\epsilon^2} \log \frac{1}{\delta}$ for a sufficiently large constant $c$ then, with probability at least $1 - \delta$,

$$\left| \mu - \frac{1}{n} \left( \sum_{i=1}^{n} X_i \right) \right| < \epsilon.$$

[Hint: Use Hoeffding's inequality from Problem 18.]

### Exercise 54

Prove that the pseudodimension of a finite set $\mathcal{C}$ of real-valued functions is at most $\log_2 |\mathcal{C}|$.

## Problems

### Problem 35

The point of this problem is to prove that, for interesting positive results for self-improving sorting algorithms, some type of restriction on the distribution over inputs (like independence of the $x_i$'s) is necessary. Throughout this problem, you can restrict attention to deterministic sorting algorithms (for simplicity), and you can assume that the array length $n$ is sufficiently large.

(a) (2 points) For a set $\mathcal{S}$ of $2^n$ permutations of $\{1, 2, \ldots, n\}$, let $D_{\mathcal{S}}$ denote the distribution that is uniform on $\mathcal{S}$ (and 0 for permutations outside $\mathcal{S}$). Explain why the entropy of $D_{\mathcal{S}}$ is $n$.

[Remark: this means that we are aspiring toward a self-improving sorter that uses $O(n)$ expected comparisons whenever the inputs are i.i.d. draws from a distribution of the form $D_{\mathcal{S}}$.]

(b) (2 points) Prove that there are at least $(n!/2^n)^{2^n}$ distinct choices of $\mathcal{S}$ and hence of $D_{\mathcal{S}}$.

(c) (2 points) Suppose that sorting algorithm $A$ uses at most $cn$ expected comparisons to sort inputs drawn from $D_{\mathcal{S}}$. Prove that $A$ correctly sorts at least half of the permutations of $\mathcal{S}$ using at most $2cn$ comparisons.

(d) (2 points) How many different permutations can a sorting algorithm correctly sort using at most $k$ comparisons?

(e) (8 points) Let $c > 0$ be an arbitrary constant (independent of $n$). Prove that if $\mathcal{A}$ is a collection of sorting algorithms such that, for every distribution of the form $D_{\mathcal{S}}$, there is an algorithm $A \in \mathcal{A}$ that requires at most $cn$ expected comparisons to correctly sort inputs drawn from $D_{\mathcal{S}}$, then $\mathcal{A}$ has size doubly exponential in $n$.

[Hint: use (c) and (d) to upper bound the number of distinct distributions of the form $D_{\mathcal{S}}$ that a single sorting algorithm can simultaneously have good performance for. Then use (b).]

(f) (4 points) Explain why (e) implies that every self-improving sorter that works for arbitrary input distributions requires space and a number of samples that is exponential in $n$.

## Problem 36

(10 points) Let $\mathcal{C} = \{ax + by + c \,:\, a, b, c \in \mathbb{R}\}$ denote the set of affine functions on the plane $\mathbb{R}^2$. Prove that the pseudodimension of $\mathcal{C}$ is precisely 3.