Exercises

Exercises are for fun and edification, please do not submit. But please do try to solve them, we may need ideas from there later in the course (even in this very homework)!

1. **DP practice.** Please solve the problems on the worksheet from 15-451.

2. **Regression.** Consider the problem of fitting a line to a set of points in the x-y plane. The points are \( \{(x_i, y_i)\}_{i=1}^{n} \). So you fit a line \( y = ax + b \), where \( a \) and \( b \) are the scalars you need to determine. The squared loss is \( \sum_{i}(y_i - ax_i - b)^2 \); minimizing this is the problem of (least-squares or \( \ell_2 \)) regression. Find an explicit formula for the optimal \( a \) and \( b \). [Check your answer against that in KT §6.3.] Hint: differentiation.

3. **(The Coffee Shoppe)** You want to open a single coffee shop on a long highway. There are \( n \) houses on the highway at locations \( x_1, x_2, \ldots, x_n \). You want your solution to be fair, so you want to locate your shop at location \( a \) to minimize \( \sum_{i} |x_i - a| \), the average distance traveled by the households. What is the optimal point \( a \) (give a one-word description)?

   Now suppose you change the objective function to \( \sum_{i} (x_i - a)^2 \). What is the new optimal solution? (Again, one word please.) What if you change it to \( \max_{i} |x_i - a| \)? Which of these is the “fairest” solution, in your opinion?

4. **(I Have the Power!)** Suppose you want to compute the matrix power \( A^t \) for some \( n \times n \) matrix \( A \) and integer \( t \). If we can multiply \( n \times n \) matrices in time \( M(n) \), it is trivial to compute \( A^t \) in \( (t-1) \cdot M(n) \) time, simply by multiplying \( A \) with itself \( t-1 \) times. Show that you can compute it in time \( O(\log t) \cdot M(n) \) time. Hint: solve this first for \( t \) being a power of 2, and then solve it for general values of \( t \).

5. **(What did you expect?)** Given a random variable \( X \) taking non-negative integer values, show that \( \mathbb{E}[X] = \sum_{i=1}^{\infty} \mathbb{P}[X \geq i] \).

6. **(Balls in Bins)** You throw \( n \) balls independently into \( n \) bins, with each ball being equally likely to fall into each bin.
   
   (a) Let random variable (r.v.) \( X_i \) denote the load of bin \( i \), i.e., the number of balls in bin \( i \). Show that \( \mathbb{E}[X_i] = 1 \) and \( \text{Var}[X_i] = \frac{n-1}{n} \).
   
   (b) Use Chebyshev’s inequality to show that the load of the most loaded bin is \( O(\sqrt{n}) \) with probability at least \( 1/2 \); i.e., \( \mathbb{P}[\max_{i \in [n]} X_i \leq c\sqrt{n}] \geq \frac{1}{2} \) for some constant \( c \).
   
   (c) Now use the Chernoff bound below to show that \( \max_i X_i \) is at most \( O(\log n) \) with probability at least \( 1 - 1/n^{100} \).
Here are the Chernoff-Hoeffding bounds we use in lecture: given independent r.v.s $Y_1, \ldots, Y_m$ taking values in $[0, 1]$, define $S := \sum_{i=1}^m Y_i$ with $\mu := \mathbb{E}[S]$. Then for any $\lambda \geq 0$,

$$\Pr[S \geq \mu + \lambda] \leq \exp(-\lambda^2/(2\mu + \lambda)) \quad (1)$$

$$\Pr[S \leq \mu - \lambda] \leq \exp(-\lambda^2/3\mu). \quad (2)$$

Please be sure to verify the independence of the random variables you apply the bound to!

7. **Simple Samplers.** Suppose $X$ is a random variable which takes on values in the interval $[0, 1]$; let $\mathbb{E}[X] = c$, and variance of $X$ be $\sigma^2$. Initially, you don’t know anything about $c$, or about the probability distribution of $X$. However, you are given a black-box that every time you query, it gives you an independent random sample drawn according to $X$. You want to estimate $c$.

A natural scheme is: sample from the black-box $N$ times—call these samples $X_1, X_2, \ldots, X_N$—and return the empirical mean $\hat{c} := \frac{1}{N} \sum_{i=1}^N X_i$. The natural question is: how big does the number of samples $N$ have to be so that

$$\Pr[|\hat{c} - c| \leq \epsilon] \geq 1 - \delta. \quad (3)$$

I.e., you want to be within error $\epsilon$ with confidence $1 - \delta$.

(a) Use Chebyshev’s inequality to show that $N = O(\frac{\sigma^2}{\epsilon^2}) = O(\frac{1}{c^2})$ samples suffice to ensure (3). Hence, to get $\delta = 1/n^k$ for some value $n$, we would take $O(n^k/\epsilon^2)$ samples.

(b) Use the Chernoff-Hoeffding bounds (given above) to show that $N = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ samples suffice. Hence, to get $\delta = 1/n^k$ we would take $O(\frac{k \log n}{\epsilon^2})$ samples, which are far fewer.

Problems

Please solve any three problems.

1. **(Independent Sets.)** Given a graph $G = (V, E)$, an subset $S \subseteq V$ of vertices is independent if no edge contains both endpoints in $S$. In the DP lecture notes we saw how to compute a max-weight independent set, when the graph is a tree. (It is NP-hard in general graphs, as we will see later.) Remember that the empty set is also independent.

   (a) Give an $O(n)$ time algorithm to compute the number of independent sets of a tree $T = (V, E)$. Make sure your algorithm computes the right answer for the tree consisting of a single edge (it should be 3) and for a tree consisting of a path of length 2 (the answer should be 5).

   (b) You want an algorithm to sample uniformly from the set of all independent sets of $T$. I.e., $\mathcal{I}_T$ is the collection of all independent sets of $T$, then the algorithm should return each $S \in \mathcal{I}_T$ with probability $\frac{1}{|\mathcal{I}_T|}$. You may assume that you have access to a random number generator $\text{Coin}(\text{real } r)$ that takes a single input $r \in [0, 1]$; it returns 1 with probability exactly $r$, and 0 otherwise. Your algorithm should run in $O(n)$ time as well, and can use quantities computed in part (a).

   (c) Now your friend in physics wants to sample from the Potts model: you are given a parameter $\lambda > 0$ and the goal is to sample each $S \in \mathcal{I}_T$ with probability

   $$\frac{\lambda^{|S|}}{Z(G)}$$
where $Z(T) = \sum_{I \in \mathcal{I}} \lambda^{|I|}$ is the “partition function” (a.k.a. the normalizing constant). Give linear-time algorithms to compute $Z(T)$ and to sample from the Potts model. *(Note that this problem generalizes the previous two, when you set $\lambda = 1$, so a correct solution for this can give you points for the above parts too.)*

(d) (Optional: 5 bonus points.) Now you want to compute, for each vertex in the graph, the number of independent sets that contain that vertex. Show how to compute all $n$ counts in linear time: quadratic time is easy if you can solve the above parts.

2. *(Options pricing.)* You start working for a company, and have to learn about finance. An *option* (specifically, an “American call option”) gives the holder the right to purchase some underlying asset (e.g., one share of ElGoog) at some specified exercise price (e.g., $100) within some specified time period (e.g., 1 year). The value of an option depends on the current price of the asset, the exercise price of the option, the length of the time period, and one’s beliefs about how the asset’s price is likely to behave in the future.

For example, take an easy case: suppose we have an option to buy one share of ElGoog at $100 that expires right now. If ElGoog is currently selling for $105, then the value of this option is $5, since we can buy a share for $100 and sell it for $105. If ElGoog is currently selling for $95, then the value of this option is $0 (we wouldn’t want to exercise it). However, suppose the option expires tomorrow, and suppose we have some simple model of how ElGoog shares behave. E.g., perhaps our model says that each day, with probability 1/4 the share price goes up by $10, and with probability 3/4 the share price goes down by $5. In that case, if ElGoog is currently worth $95, then the value of this option is $1.25 because that is our expected gain if we use the optimal strategy “wait until tomorrow, and then exercise the option if ElGoog went up” (our expected gain is $\frac{1}{4} \times 5 + \frac{3}{4} \times 0$). If ElGoog is currently worth $105, then the value of the option is $5 (because if you work it out, our optimal strategy is to exercise the option right away).

Formally, the value of an option is its expected value under the optimal strategy for using it, given our probabilistic model for the stock. Note that the optimal strategy for using the option need not commit in advance to what day the option will be exercised, and the date it gets exercised (if ever) may depend on how the stock has performed so far.

Assume stock prices are integers between 0 and $B$. Suppose we are given the following simple probabilistic model $P_{ij}$ for how the stock behaves: specifically, if the stock has price $i$ on day $t$, then the probability that the stock will have price $j$ on day $t+1$ is $P_{ij}$. So, for each $i$, $\sum_j p_{ij} = 1$.

(a) Give a dynamic-programming algorithm to calculate the value of an option of exercise price $X$ that expires $T$ days in the future, given that the current price of the stock is $S$. The running time of your algorithm should be $O(B^2 T$).

(b) Suppose our option can only be exercised at exactly time $T$, rather than any time $\leq T$ (this is called a “European option”). Describe an algorithm to solve for the option’s value that runs in time $O(B^3 \log T)$.

3. *(Counting Unique Elements.)* Suppose we are given a positive integer $K$, and we want to estimate the number of distinct elements in a stream $a_1, a_2, \ldots, a_n$. In particular, we want to find out if the number of distinct elements is at most $K$ (“low”), or at least $2K$ (“high”). Here is an algorithm:
• Fix a 2-wise independent hash family $H : U \rightarrow \{0,1,2,\ldots,4K-1\}$, and pick a hash function $h$ from it.
• If for some $t \in \{1,2,\ldots,n\}$, we have $h(a_t) = 0$, say “High”; else say “Low”. (So, we don’t need to keep a hash table to run this algorithm, just a flag that tells us if anything has hashed to 0.)

(i) Show that if the number of distinct elements is at most $K$, then
\[
\Pr[\text{we say “High”}] \leq 1/4.
\]

(ii) Show that if the number of distinct elements is at least $2K$, then
\[
\Pr[\text{we say “High”}] \geq 3/8.
\]

(Hint: Show this for the case when the number of distinct elements is $2K$. How does the probability behave when this number increases.)

(iii) Now to boost the success probability, we consider $m$ copies of the above data structure, with independently chosen hash functions. Some of these might say “High” and some say “Low”. If the fraction of Highs is larger than average($\frac{1}{4}, \frac{3}{8}$) = $\frac{5}{16}$ then we say “High”, else we say “Low”. Prove that the probability that this boosted data structure makes a mistake is at most $2e^{-cm}$ for some universal constant $c \in [0,1]$.

Hint: Inclusion-exclusion (or rather its probabilistic counterpart). Also, Chernoff-Hoeffding.

4. (Fast Probing.) Recall from class that linear probing is the following simple scheme for resolving hash collisions: we use a hash function $h : U \rightarrow [M]$. If an element hashes to a cell that already contains something, search to the right (wrapping around at the end) until a free cell is found, and place the element there. Similarly, to query an element $q$, start with the cell $h(q)$, and scan to the right (wrapping around the end) until you find the element $q$ or an empty cell. We now show that for a totally random hash function $h$, the expected time for a find/insert is $O(1)$ (assuming no deletions).

Suppose we insert a total of $n$ elements into a table of size $M = 4n$. A run is a maximal interval of occupied table cells. The insert/find time for element $e$ is bounded by the length of the run containing the location $h(e)$, so we want to understand the probability of long runs occurring.

Purely for sake of the analysis, we think of the cells of the hash table as being the leaves of an imaginary complete binary tree of height $\log_2 M$ (where the leaves are at height 0). Define a node $v$ at height $k$ in this binary tree to be suspicious if at least $1/2 \cdot 2^k$ elements hash to a leaf in the subtree $T_v$ under $v$.

(a) For a node $v$ at height $k$, how many elements do you expect to hash into the subtree $T_v$? Show that probability of $v$ being suspicious is $e^{-c2^k}$ for some constant $c \in (0,1)$.

(b) Consider a run $R$ whose length lies in the range $[2^\ell,2^{\ell+1})$ for $\ell \geq 3$. Now consider the set of nodes $v$ at height $k = \ell - 2$ in the imaginary tree such that the leaves of the subtree rooted at each $v$ intersect with $R$. For example in the picture below this is $\{a,b,c,d\}$.
What is the minimum and maximum size of this set? Argue that among these nodes at height \( k = \ell - 2 \), at least one must be suspicious.

(c) Use the previous parts to bound the probability that the run containing any given cell \( x \) has length in \([2^\ell, 2^{\ell+1})\). Conclude that the expected length of the run containing any given cell \( x \) is \( O(1) \).