1. **(14 pt.) [Another way to sketch sparse vectors.]** Suppose that A is an list of length n, containing elements from a large universe $\mathcal{U}$. Our goal is to estimate the frequencies of each element in $\mathcal{U}$: that is, for $x \in \mathcal{U}$, how often does x appear in A?

   The catch is that A is too big to look at all at once. Instead, we see the elements of A one at a time: $A[0], A[1], A[2], \ldots$. Unfortunately, $\mathcal{U}$ is also really big, so we can't just keep a count of how often we see each element.

   In this problem, we'll see a construction of a randomized data structure that will keep a "sketch" of the list A, use small space, and will be able to efficiently answer queries of the form "approximately how often did x occur in A"?

   Specifically, our goal is the following: we would like a (small-space) data structure, which supports operations `update`(x) and `count`(x). The `update` function inserts an item $x \in \mathcal{U}$ into the data structure. The `count` function should have the following guarantee, for some $\delta, \epsilon > 0$. After calling `update` n times, `count`(x) should satisfy

   $$C_x \leq \texttt{count}(x) \leq C_x + \epsilon n \tag{1}$$

   with probability at least $1 - \delta$, where C_x is the true count of x in A.

   (a) **(3 pt.)** Your friend suggests the following strategy (this will not be our final strategy). We start with an array R of length b initialized to 0, and a random hash function $h : \mathcal{U} \to \{0, 1, \ldots, b-1\}$. You can assume that h is drawn from some universal hash family, i.e $P(h(x) = h(y)) = 1/b$ for any $x \neq y$. Then the operations are:

   - `update`(x): Increment $R[h(x)]$ by 1.
   - `count`(x): return $R[h(x)]$.

   For every entry $A[i]$ in the list it encounters, the scheme calls `update`$(A[i])$.

   After sequentially processing all n items in the list, what is the expected value of `count`(x)?

   (b) **(2 pt.)** Show that there is a choice of b that is $O(1/\epsilon)$ so that, for any fixed $x \in \mathcal{U}$, we have

   $$\Pr[\texttt{count}(x) < C_x] = 0$$

   and

   $$\Pr[\texttt{count}(x) \geq C_x + \epsilon n] \leq \frac{1}{e}.$$

   [**HINT:** *The first of the requirements is true no matter what b is.*]

   (c) **(2 pt.)** Explain how you would use T copies of the construction in part (a) to define a data structure that, for any fixed $x \in \mathcal{U}$, satisfies (1) with high probability. How big do you need to take T so that the (1) is satisfied with probability at least $1 - \delta$? How much space does your modified construction use? (It should be sublinear in $|\mathcal{U}|$ and n). Give a complete description and analysis of the data structure, and explain how much space it uses. You may assume that it takes $O(\log |\mathcal{U}|)$ bits to store the hash function h and $O(\log n)$ to store each element in the array R.

(d) Explain how to use your algorithm to solve the following problem:

i. **(4 pt.)** Given a k-sparse vector $a \in \mathbb{Z}^N_{\geq 0}$ ($\mathbb{Z}_{\geq 0}$ is the set of non-negative integers), design a randomized matrix $\Phi \in \mathbb{R}^{m \times N}$ for $m = O(\frac{k \log N}{\epsilon})$ so that the following happens. With probability at least 0.99 over the choice of Φ, you can recover $\tilde{a}$ given Φa, so that simultaneously for all $i \in 1, ..., N$, we have

$$|\tilde{a}[i] - a[i]| \leq \frac{\epsilon \|a\|_1}{2k}.$$

[**HINT:** *Think of the k-sparse vector a as being the histogram of the items in the list A from the previous parts.*]
[**HINT:** *How can you represent a hash function as a matrix multiplication?*]
[**HINT:** *Note that we want a tighter bound, and we want the bound to hold simultaneously for all i. How can we change b and T to achieve this?*]

ii. **(3 pt.)** Now, assuming the above holds for all i, use the k-sparseness of a to construct $\hat{a}$ from $\tilde{a}$ such that

$$\|\hat{a} - a\|_1 \leq \epsilon \|a\|_1.$$

iii. **(0 pt.) [This question is zero points, but worth thinking about.]** How does the guarantee in the previous part compare to the RIP matrices (and the compressed sensing guarantee that we can get from them, Theorem 1 in the Lecture 9 lecture notes) that we saw in class? (i.e., is this guarantee weaker? Stronger? Incomparable? The same?)

SOLUTION:

(a) Let the set $S \subset \{1, 2, ..., n\}$ denote the indices i such that $A[i] = x$. Then clearly $|S| = C_x$.
In the i-th `update` step, $R[h(x)]$ will be incremented if

- $A[i] = x$, or
- $A[i] \neq x$ but $h(A[i]) = h(x)$

We thus have $\texttt{count}(x) = \sum_{i \in S} 1 + \sum_{i \notin S} 1_{h(A[i])=h(x)} = C_x + \sum_{i \notin S} 1_{h(A[i])=h(x)}$. Using linearity of expectation, and the fact that $\Pr(h(A[i]) = h(x)) = 1/b$ for any $A[i] \neq x$, we get $\mathbb{E}[\texttt{count}(x)] = C_x + \frac{|[n] \setminus S|}{b} = C_x + \frac{n - C_x}{b}$.

(b) Since $\texttt{count}(x) = C_x + \sum_{i \notin S} 1_{h(A[i])=h(x)}$ and indicator random variables are always non-negative, we have $\texttt{count}(x) \geq C_x$ as always true. This means the random variable $Y := \texttt{count}(x) - C_x$ is always positive, so we can apply Markov's inequality to it. From part (a), we know that $\mathbb{E}[Y] = \frac{n - C_x}{b} \leq n/b$.
Using Markov's inequality, this gives $\Pr[\texttt{count}(x) - C_x \geq \epsilon n] \leq \frac{1}{b\epsilon}$. Thus, taking $b = O(\frac{1}{\epsilon})$ with some large enough constant suffices.

(c) Fix any $x \in \mathcal{U}$. Consider a strategy that repeats the scheme from part (a) T times in parallel for some large integer T that we will choose later. For each such scheme, denote the corresponding counter as $\texttt{count}^i(x)$ for $i = 1, 2, ..., T$. Then let our overall scheme output $\texttt{count}(x) = \min_{i=1,2,...,T} \texttt{count}^i(x)$ as our estimate for C_x. Clearly, $\texttt{count}(x) \geq C_x$. It is important that we use a separate independently chosen hash function for each of these T schemes.

For a single scheme from part(a), the space usage is $O(\frac{\log n}{\epsilon} + \log |\mathcal{U}|)$ because we need to store an array of length $b = O(\frac{1}{\epsilon})$ and one hash function. For our overall scheme, since we do T such things in parallel, the space usage is $O(T \cdot (\frac{\log n}{\epsilon} + \log |\mathcal{U}|))$ bits.

Note that since $\texttt{count}(x) = \min_{i=1,2,...,T} \texttt{count}^i(x)$, $\texttt{count}(x) \geq C_x + \epsilon n$ if and only if $\texttt{count}^i(x) \geq C_x + \epsilon n$ for all $i = 1, ..., T$. Because the randomness used in each scheme is independent (the hash functions are chosen independently), using the result of part (b) we notice that such a failure happens with probability at most $\frac{1}{e^T}$. Setting $T = \Theta(\log \frac{1}{\delta}))$ with some large enough constant ensures $\frac{1}{e^T} \leq \delta$.

Thus the total space usage is $O(\log(\frac{1}{\delta}) \cdot (\frac{\log n}{\epsilon} + \log |\mathcal{U}|))$ bits.

(d) i. If we interpret a as the counts of $||a||_1$ items, we can represent the scheme from part (c) as a multiplication by a random matrix $\Phi \in \mathbb{R}^{m \times N}$, where $m = bT$, to encode a. Note that we can represent a random hash function $h_i : \{0, ..., N-1\} \to \{0, ..., b-1\}$ as a matrix $H_i \in \mathbb{R}^{b \times N}$, by setting

$$H_i[\ell, m] = 1_{h_i(m)=\ell}$$

Then, $H_i a$ will contain in each index the sum of the counts of all the elements that were hashed to that specific index, which is exactly the construction in part(a). Note that the analysis from part (b) also carries through. We can use Φ to represent T hash functions by stacking the matrices H_i on top of each other. We can construct $\tilde{a}[i]$ from Φa using the same strategy as part (c), by taking the minimum value over the T indices that i was hashed to. By taking $b = O(\frac{k}{\epsilon})$ we can guarantee that with $T = 1$, we have

$$P(|\tilde{a}[i] - a[i]| \geq \frac{\epsilon \|a\|_1}{2k}) \leq \frac{1}{e}$$

By taking $T = \Theta(\log N)$, this becomes

$$P(|\tilde{a}[i] - a[i]| \geq \frac{\epsilon \|a\|_1}{2k}) \leq \frac{1}{N^2}$$

Taking a union bound gives the desired result

ii. **Approach 1** Define

$$\hat{a}[i] = \begin{cases} \tilde{a}[i] & \tilde{a}[i] > \frac{\epsilon \|a\|_1}{2k} \\ 0 & \tilde{a}[i] \leq \frac{\epsilon \|a\|_1}{2k} \end{cases}$$

if $a[i] = 0$, we have $\tilde{a}[i] \leq \frac{\epsilon \|a\|_1}{2k}$, and thus $\hat{a}[i] = 0$. If $a[i] \neq 0$, we know $|\tilde{a}[i] - a[i]| \leq$

$\frac{\epsilon\|a\|_1}{2k}$ and $|\hat{a}[i] - \tilde{a}[i]| \leq \frac{\epsilon\|a\|_1}{2k}$. Thus, $|\hat{a}[i] - a[i]| \leq 2\frac{\epsilon\|a\|_1}{2k}$. Therefore,

$$\begin{aligned}\|\hat{a} - a\|_1 &= \left(\sum_{a[i]=0} |\hat{a}[i] - a[i]|\right) + \left(\sum_{a[i]\neq 0} |\hat{a}[i] - a[i]|\right) \\ &\leq 0 + k \times \left(\frac{\epsilon \|a\|_1}{k}\right) = \epsilon \|a\|_1 .\end{aligned}$$

Approach 2 Consider the following strategy: construct $\hat{a}$ by setting all but the largest k values in $\tilde{a}$ to 0. We claim that $\hat{a}$ has the desired property. First, note that if the nonzero entries in $\hat{a}$ are the nonzero entries in a, then we have error at most $k\frac{\epsilon\|a\|_1}{2k} \leq \epsilon\|a\|_1$.
Now, suppose we made $t \leq k$ errors. That is, some indices $r_1, .., r_t$ are nonzero in a but have been set to 0 in $\hat{a}$. This implies there are indices $s_1, ..., s_t$ that are zero in a which were not set to 0 in $\hat{a}$. First, note that by the bound from the first part, we have

$$\tilde{a}[s_j] - a[s_j] = \tilde{a}[s_j] \leq \frac{\epsilon\|a\|_1}{2k}$$

In addition, for r_i to be set to 0 and s_j to not be set to 0, we must have

$$\tilde{a}[r_i] \leq \tilde{a}[s_j]$$

This implies

$$a[r_i] \leq \tilde{a}[r_i] \leq \tilde{a}[s_j] \leq \frac{\epsilon\|a\|_1}{2k}$$

Noting that $\hat{a}[r_i] = 0$ and $\hat{a}[s_j] = \tilde{a}[s_j]$, we have the following

$$|\hat{a}[r_i] - a[r_i]| = |a[r_i]| \leq \frac{\epsilon\|a\|_1}{2k}$$
$$|\hat{a}[s_j] - a[s_j]| = |\tilde{a}[s_j]| \leq \frac{\epsilon\|a\|_1}{2k}$$

Therefore, the total error is bounded by

$$2t\frac{\epsilon\|a\|_1}{2k} \leq \epsilon\|a\|_1$$

iii. The guarantee is similar. In particular, both give us a matrix with asymptotically the same number of rows (if we treat ϵ as constant), and both give us bound on the ℓ_1 error.
This guarantee is different in a few ways. Here are some ways that it is weaker:

- The form is "for all a, w.h.p. over Φ....", while the compressed sensing guarantee is (assuming we have a randomized construction of Φ) "w.h.p. over Φ, for all a, ..."
- In the compressed sensing guarantee, we actually get *zero* error, since the right hand side, $\|x - x_k\|_1$, is equal to zero when x is exactly k sparse. Here we do get a little bit of error that depends on ϵ.

However, this construction does give us better algorithms:

- The recovery algorithm here takes time $\tilde{O}(N)$, since for each item in the universe $\mathcal{U}$, we just use our sketch to estimate it. In contrast, the recovery algorithm from the RIP involves solving an LP, which takes much longer (though still polynomial in N). In fact, it is possible to adapt the recovery algorithm for the construction in this problem to be $O(k \log^2(N))$! [Fun exercise: figure out how!]. So that's much nicer.
- We also have the ability to query elements of $\hat{a}$ in $O(\log N)$ time for this construction, which we don't get from the RIP solution.

2. **(6 pt.)[Bayesians and Frequentists.]**

   Suppose that there are n statisticians, with each one being either a "Bayesian" or a "Frequentist". We have a bunch of math problems to solve, and suppose that each statistician has a list of at least $1 + \log_2(n)$ problems that they would be happy to work on. The statisticians' lists may be different and may overlap—it could even be the case that everyone has the same list.

   Unfortunately, "Bayesians" and "Frequentists" sometimes have trouble collaborating.

   Use the probabilistic method to show that it is possible to assign the math problems to the statisticians such that 1) each statistician is assigned a problem to work on, and 2) for each problem, either it is unassigned, it is assigned to only Bayesians, or it is assigned to only Frequentists [ie we won't need to worry about collaboration squabbles]. Note that it is fine to assign lots of people to a single problem, as long as all the people assigned are the same type, and it is fine for some problems to be unassigned.

   [**HINT:** *To use the probabilistic method, you just have to define a probability distribution over assignments such that with* positive *probability, the assignment satisfies the criteria. A first attempt might be to assign each statistician to a random problem from their list of problems—this distribution does* not *seem to work...you'll have to come up with a more interesting distribution.*]

   SOLUTION: Consider tossing an independent coin for each problem, and assigning that problem to "Bayesians" if the coin lands heads, and otherwise assign the problem to "Frequentists". Since each of the n peoples' lists have size $\geq 1 + \log_2 n$, for each person the probability that no problem on their list is assigned to their "type" is at most $1/2^{1+\log_2 n} = 1/2n$. Hence, by a union bound over the n people, there is a positive probability that everyone has a problem on their list assigned to their type. Hence the probabilistic method guarantees that an acceptable assignment must exist.

3. **(12 pt.)[Who needs geometry when you have the probabilistic method?]**

   Suppose that your friend has – perhaps adversarially – drawn k points on the surface of a flat table, which you can think of as an infinite plane. Your goal will be to cover all the

points with pennies, *without any pennies overlapping each other*. How big can k such that it is *always* possible for you to cover these k points with non-overlapping pennies? [Think of pennies as circles of radius 1.]

(a) **(1 pt.)** Prove that there is a way of placing non-overlapping pennies on the plane such that the pennies cover more than 90% of the area of the plane. [Hint: the optimal configuration is the hexagonal tiling used to arrange oranges in the supermarket. The precise area covered in this optimal configuration will be the following fraction: $\pi/(2\sqrt{3}) \approx 0.907$.]

(b) **(4 pt.)** Prove that there is some configuration of $k = 1000$ points such that it is impossible to cover them all with non-overlapping pennies.

(c) **(0 pt.)** What is the smallest value of k for which you can prove that there is a configuration that is impossible to cover?

(d) **(6 pt.)** Prove that for any configuration of 10 points, its possible to cover them via at most 10 non-overlapping pennies. [Hint: Use part (a) and the probabilistic method!]

(e) **(1 pt.)** Would your probabilistic proof work for a set of 11 points? Why or why not?

(f) **(0 pt.)** Can you come up with a non-probabilistic method proof for the fact that any 10 points can be covered with non-overlapping pennies?

SOLUTION:

(a) This is just a calculation—the easiest way is to take tile the plane with equilateral triangles of edge length 2 (where the radius of each penny is 1 unit), and note that the fraction of the area of the triangle covered by pennies located at each of the corners is exactly the claimed number.

(b) There are many possible proofs—we'll give full credit to any reasonable description that could be made rigorous, even if your "proof" is a bit hand-wavy. One high-level idea is to show that any configuration of pennies must always will contain a small uncovered circle, then argue that if you have a grid of, say, 1000 points spaced closely together, at least one such point will lie in the uncovered circle.

(c) I *think* I can prove that 16 equally spaced points around a circle of radius 1.0001 cannot be covered by non-overlapping pennies of radius 1. Proof might also work for 15 though haven't checked :)

(d) Consider taking an infinite sheet of pennies arranged in the hexagonal lattice that covers $> 90\%$ of the plane, and putting it down in a uniformly random orientation. For each of our 10 points, the probability it is *not* covered is at most $1 - 0.907 < 0.093$, hence by a union bound over the 10 points, the probability that any of the 10 pennies are NOT covered is at most $10 \cdot 0.093 = 0.93 < 1$. Hence there is a positive probability (about 7%) that we have covered all 10 pennies, which means that there must exist a configuration of this infinite hexagonal lattice of pennies that does cover all 10 pennies, by the probabilistic method. Given such a configuration of pennies, since there are only 10 points, if we remove any penny that doesn't cover a point, we will be left with at most 10 pennies (at most 1 penny per point).

(e) The probabilistic proof above would not work, since the probability each point is covered is $\approx 0.907 < 1 - 1/11$. A slightly more clever probabilistic argument does work though...