Problems 1, 2 require you to have read the lecture notes or watched the lecture, but you should be able to attempt them on your own after that. Similar for 3 and 4.

Problem 5 takes some time, but is important to check you understand how the Morris counter works. The last subquestion is useful to make you think about how to implement some general building blocks requiring sampling bias bits, but a little bit harder.

Problem 6 is short, once you have the right idea. You can skip it if you don't see how to easily solve it, but read the solution later.

Problem 7 is quite involved, but useful to have seen (it introduces an interesting, "well known" algorithm): it is fine to skip it, but in that case still have a look at the algorithm (and the solution) afterwards.

Warm-up

Problem 1. Go through the "median-of-means" proof for the Morris counter, to prove the statement about the result.

Solution 1. Given that

$$\mathbb{E}[C_n] = n+1 \text{ and } \mathrm{Var}[C_n] = \frac{n(n-1)}{2},$$

the main issue is that $\mathbb{E}[C_n]^2$ is at the same order as $\mathrm{Var}[C_n]$, which will makes it hard for Chebyshev's inequality to get good additive estimate – you get probability bound of the form

$$\frac{\mathrm{Var}[X]}{(\varepsilon\mathbb{E}[X])^2} = \frac{1}{\varepsilon^2},$$

and for $\varepsilon \leq 1$, it is a vacuous bound.

Taking a means with repetition of size k can effective bring down the variance by a factor of $1/k$ while keeping the mean unchanged. This allows us to get (by Chebyshev) a $(1+\varepsilon)$-factor estimate with large (constant) probability: once we have this, we can use the median trick to boot the constant probability of success to $1-\delta$.

Problem 2. Instead of using the "median-of-means" trick to boost the accuracy of the Morris counter, what happens if we were to do the opposite and use the "mean-of-medians"?

Solution 2. It will not help as the probability that you get good estimate is low to begin with. So you "expect" to see a lot of bad estimates in your repetition. Taking a median out of them gives no good statistical guarantee.

Problem 3. If we were to use truly random hash functions instead of a strongly universal hash family in the BKJST algorithm, what would the space complexity become?

Solution 3. A truly random hash function takes $n \log n$ bits to store (compared to one for a strongly universal hash family's $O(\log n)$).

Problem 4. Check your understanding: why are we using a hash function g in the BKJST algorithm? What would happen if we were to store j in the bucket B, instead of $g(j)$?

Solution 4. It is to save space. The alternative would be to store j directly and doing so would take up $O(\log n \cdot |B|) = O(\frac{\log n}{\varepsilon^2})$ bits!

Problem solving

Problem 5. Analyse the "careful variant" of the Morris counter, where instead of doubling C with probability $1/C$, we multiply C by $1+\alpha$ with probability $1/(\alpha C)$.

a) Compute the expectation of C at the end of the algorithm.

b) Compute its variance, and conclude by Chebyshev.

c) Explain how you would set α to get a $(1+\varepsilon)$-factor estimate of the true count with probability at least $1-\delta$ using the median trick, and give the resulting space complexity, *almost* proving the theorem stated in the lecture (but with a multiplicative instead of additive $\log(1/\delta)$.

d) Explain how you would set α to get a $(1+\varepsilon)$-factor estimate of the true count with probability at least $1-\delta$ *without* using the median trick, and give the resulting space complexity, *actually* proving the theorem stated in the lecture (with an additive $\log(1/\delta)$).

e) ($\star$) How would you actually implement the increment step (i.e., how, given random uniform bits, would you "multiply C by $1+\alpha$ with probability $1/(\alpha C)$")?

Solution 5.

a) Denote Z_i the indicator random variable for the event that C_i is increased by a factor of $(1+\alpha)$, with probability $1/\alpha C_i$. So,

$$C_{i+1} = (1+\alpha Z_i)C_i.$$

Plugging into the computation for expectation,

$$\mathbb{E}[C_{i+1}] = \mathbb{E}[\mathbb{E}[C_{i+1} \mid C_i]] = \mathbb{E}[\mathbb{E}[(1+\alpha Z_i)C_i \mid C_i]].$$

As C_i is fixed in the inner (conditional) expectation, $Z_i \sim \text{Bern}\left(\frac{1}{\alpha C_i}\right)$, and thus,

$$\mathbb{E}[(1+\alpha Z_i)\cdot C_i \mid C_i] = \mathbb{E}\left[\left(1+\alpha\cdot\frac{1}{\alpha C_i}\right)\cdot C_i \mid C_i\right] = C_i + 1.$$

Since $\mathbb{E}[C_0] = 1$, we can show that (by induction),

$$\mathbb{E}[C_n] = n+1.$$

For the variance, we follow the same idea: first, $\text{Var}[C_n] = \mathbb{E}[C_n^2] - \mathbb{E}^2[C_n]$, and so all we need to do is to compute $\mathbb{E}[C_n^2]$. To do so, rewrite

$$\mathbb{E}[C_n^2] = \mathbb{E}[\mathbb{E}[C_{i+1}^2 \mid C_i]] = \mathbb{E}_{C_i}[\mathbb{E}[(1+\alpha Z_i)^2 C_i^2 \mid C_i]].$$

Again, as C_i is fixed in the inner expectation, and since $Z_i \sim \text{Bern}\left(\frac{1}{\alpha C}\right)$ (so that $Z_i^2 = Z_i$), $\mathbb{E}[Z_i^2 \mid C_i] = \mathbb{E}[Z_i \mid C_i] = \frac{1}{\alpha C_i}$, and so

$$\mathbb{E}[(1+2\alpha Z_i + \alpha^2 Z_i^2)C_i^2 \mid C_i] = C_i^2 + (2+\alpha)\cdot C_i.$$

Noting that $\mathbb{E}[C_i] = i+1$ from before, and as $\mathbb{E}[C_0^2] = 1$, we have

$$\mathbb{E}[C_n^2] = \mathbb{E}[C_{n-1}^2] + (2+\alpha)\cdot((n-1)+1) = \mathbb{E}[C_{n-2}^2] + (2+\alpha)\cdot(n+n-1).$$

Unrolling this further, we get that

$$\mathbb{E}[C_n^2] = 1 + (2+\alpha)\cdot\sum_{i=0}^{n-1}(i+1) = 1 + \frac{(2+\alpha)\cdot n(n+1)}{2},$$

and so

$$\text{Var}[C_n] = \mathbb{E}[C_n^2] - \mathbb{E}^2[C_n] = 1 + \frac{(2+\alpha)\cdot n(n+1)}{2} - (n+1)^2 = \frac{\alpha}{2}\cdot n(n+1) - n.$$

We see that

$$\boxed{\mathbb{E}[C_n - 1] = n} \text{ and } \boxed{\text{Var}[C_n - 1] = \frac{\alpha}{2}\cdot n(n+1) - n.}$$

b) By Chebyshev, we have

$$\Pr[|C_n - 1 - n| \geqslant \varepsilon n] \leqslant \frac{\mathrm{Var}[C_n - 1]}{\varepsilon^2 n^2},$$

and we want the RHS to be at most δ. Solving the inequality gives,

$$\frac{\alpha}{2} \cdot n(n+1) - n \leqslant \varepsilon^2 \delta n^2 \Leftrightarrow \alpha \leqslant \frac{2(\varepsilon^2 \delta n + 1)}{(n+1)} \Leftarrow \alpha = 2\varepsilon^2 \delta \leqslant \frac{2(\varepsilon^2 \delta n + 1)}{(n+1)}.$$

Plug in $\delta = \frac{1}{4}$, so that setting $\alpha = \frac{\varepsilon^2}{2}$ is enough. We will use the median trick on this configuration (run $\log \frac{1}{\delta}$ copies and take the median of them). This gives space complexity (we return to the exponent x, we know that $(1+\alpha)^x = C_n \leqslant m$):

$$\begin{aligned} O\left(\left(\log \frac{1}{\delta}\right) \cdot \log_2 \log_{1+\alpha} m\right) &= O\left(\log \frac{1}{\delta} \cdot \left(\log_2 \frac{\log_2 m}{\log_2(1+\alpha)}\right)\right) \\ &= O\left(\log \frac{1}{\delta} \cdot \left(\log \log m + \log \frac{1}{\varepsilon}\right)\right). \end{aligned}$$

c) Set $\alpha = 2\varepsilon^2 \delta$. The space complexity becomes (using $\log(1+2x) \geqslant x$, when $x \in (0,1]$)

$$\log_{(1+\alpha)} n = \frac{\log_2 n}{\log_2(1+\alpha)} \leqslant \frac{\log_2 n}{\varepsilon^2 \delta}.$$

$(1+\alpha)^x$, storing and return $(1+\alpha)^x - 1$ at the end suffices. We can cap x's growth to

$$\log_{(1+\alpha)} m \leqslant \frac{\log_2 m}{\varepsilon^2 \delta},$$

as $n \leqslant m$. So, storing x would take at most

$$\log_2 \left(\frac{\log_2 m}{\varepsilon^2 \delta}\right) = \log \log m + 2 \log \frac{1}{\varepsilon} + \log \frac{1}{\delta}$$

bits.

d) To sample from Bern(p) for $p \in [0,1]$. There is one generic approach that works, though it takes an unbounded number of uniformly random bits.

```
1: Sample $x$ from Uniform$(0,1)$.
2: If $x < p$, return 1. Otherwise return 0.
```

Of course, this comes with two issues: first, sampling exactly a uniformly random continuous variable on $[0,1]$ requires infinitely many uniformly random bits. Second, if p is not rational, it's not even clear how to represent it on a finite-precision computer, let alone how to store it.
Instead, consider first rounding α to a rational number α' such that $\alpha' \leq \alpha < 2\alpha'$: this is always possible, by density of the rationals. We will then run the algorithm with paramater α' instead of α: this only improves the error bound as $\alpha' \leq \alpha$, and does not change the space complexity by more than an additive $O(1)$, as $\log \frac{1}{\alpha'} \leq \log \frac{2}{\alpha} = 1 + \log \frac{1}{\alpha}$.
Since α' is a rational number and C a power of $(1+\alpha')$, the Bernoulli paramater p_i at step i is of the form

$$p_t = \frac{1}{\alpha' C} = \frac{1}{\alpha'(1+\alpha')^{\ell}}$$

for some $\ell \geq 0$. This means p_t is also a rational, since

$$\alpha'(1+\alpha')^{\ell} = \sum_{j=0}^{\ell} \binom{j}{\ell} \alpha^{j+1}$$

is a sum of $(\ell+1)$ rational numbers (each being an integer power of a rational number, times an integer). But sampling from a Bernoulli with rational parameter can be done efficiently: if $p = \frac{a}{b}$ with $a \leq b$, it suffices to draw X uniformly in $\{1, \ldots, b\}$, and output 1 iff $X \leq a$.

Problem 6. Given a stream σ of length m and $\varepsilon \in [0,1]$, let

$$H_{\varepsilon}(\sigma) = \{j \in [n] : f_j \geq \varepsilon \cdot m\}$$

denote the set of *ε-heavy hitters* of σ. Modify the Misra–Gries algorithm to make it output a set $H \subseteq [n]$ such that $H_{\varepsilon}(\sigma) \subseteq H \subseteq H_{\varepsilon/2}(\sigma)$. *(That is, the algorithm outputs every ε-heavy hitter, and everything it outputs is at least an $(\varepsilon/2)$-heavy hitter.)* Your algorithm should be one-pass, and use space $O(\log(mn)/\varepsilon)$.

Solution 6. Run algorithm parameter with $k = 2/\varepsilon$. Whatever remains, pick the ones with $\hat{f}_i \geqslant \frac{\varepsilon}{2} \cdot m$ to form H.

If $i \in H_{\varepsilon}(\sigma)$, then $f_i \geqslant \varepsilon \cdot m$, which implies

$$\frac{\varepsilon}{2} \cdot m \leqslant f_i - \frac{\varepsilon}{2} \cdot m \leqslant \hat{f}_i.$$

So i will be in H by how we form H.

On the other hand, if $i \in H$, then $\widehat{f}_i \geqslant \frac{\varepsilon}{2} \cdot m$, which implies $f_i \geqslant \hat{f}_i \geqslant \frac{\varepsilon}{2} \cdot m$. So $i \in H_{\varepsilon/2}(\sigma)$ by definition.

The overall space complexity is

$$O(k \cdot \log(mn)) = O\left(\frac{\log(mn)}{\varepsilon}\right).$$

Problem 7. Consider the following "Bottom-k" algorithm for the Distinct Elements problem, where $k \geq 1$ is a parameter.

```
1: Pick a hash function $h\colon [n] \to [0,1]$ from a strongly universal hash family
   ▷ Technically, from $h\colon [n] \to \{0, 1/N, 2/N \dots, 1\}$ where $N = poly(n)$ is large
   enough to not have to worry about collisions.
2: Set $z = (1, 1, \dots, 1) \in \mathbb{R}^k$
3: for all $1 \leq i \leq m$ do
4:     Get new element $a_i \in [n]$ of the stream
5:     $z \leftarrow k$ smallest values among $z_1, \dots, z_k, h(a_i)$
6: return $\hat{d} \leftarrow \frac{k}{\max(z_1,\dots,z_k)}$
```

a) Show that, for $k = \Theta(1/\varepsilon^2)$, the above algorithm returns $(1 \pm \varepsilon)$-estimate of the number of distinct elements d, with probability at least 99%. (To do so, define X_i as the indicator that $h(a_i) < \frac{k}{(1+\varepsilon)d}$, and use Chebyshev on $\sum_i X_i$.)

b) What is the space complexity of the algorithm?

Solution 7. a) Throughout the proofs, we assume $\varepsilon < 1$. Denote for $i \in \text{Distinct}(a_1, \dots, a_m)$, where $a_j \in [n]$, $X_i = \mathbb{1}_{\left\{h(a_{j_i}) < \frac{k}{(1+\varepsilon)d}\right\}}$. a_{j_i} denotes the actual value a_j takes or $a_j = i$.

$$\begin{aligned}
&\sum_{i=1}^{d} X_i > k \\
\Leftrightarrow\quad &\text{at least } k \text{ (out of } d \text{ distinct points) of them have } h(i) < \frac{k}{(1+\varepsilon)d} \\
\Leftrightarrow\quad &\max(z_1, \dots, z_k) < \frac{k}{(1+\varepsilon)d} \\
\Leftrightarrow\quad &\frac{k}{\max(z_1, \dots, z_k)} > (1+\varepsilon)d.
\end{aligned}$$

So,

$$\sum_{i=1}^{d} X_i > k \Leftrightarrow \hat{d} > (1+\varepsilon)d.$$

It remains to analyse $\Pr\left[\sum_{i=1}^{d} X_i > k\right]$. Since it is strongly universal hash family, it is marginally "uniform".[1]

$$\mathbb{E}\left[\sum_{i=1}^{d} X_i\right] = \sum_{i=1}^{d} \mathbb{E}[X_i] = \sum_{i=1}^{d} \Pr\left[h(i) < \frac{k}{(1+\varepsilon)d}\right] \cdot 1 = \frac{k}{1+\varepsilon}.$$

Also we have **pairwise independence** as we are using a strongly universal hash family, and $\mathbb{E}[X_i^2] = \mathbb{E}[X_i]$ (indicator r.v.), so that

$$\mathrm{Var}\left[\sum_{i=1}^{d} X_i\right] = \sum_{i=1}^{d} \mathrm{Var}[X_i] = \sum_{i=1}^{d} (\mathbb{E}[X_i^2] - \mathbb{E}[X_i]^2) \leqslant \frac{k}{1+\varepsilon}.$$

By Chebyshev,

$$\Pr\left[\sum_{i=1}^{d} X_i > (1+\alpha)\frac{k}{1+\varepsilon}\right] < \frac{\mathrm{Var}\left[\sum_{i=1}^{d} X_i\right]}{\alpha^2 \mathbb{E}^2\left[\sum_{i=1}^{d} X_i\right]} \leqslant \frac{\frac{k}{1+\varepsilon}}{\alpha^2\left(\frac{k}{1+\varepsilon}\right)^2} \leqslant \frac{1}{6}.$$

Let $\alpha = \varepsilon$ and solve the inequality,

$$\frac{\frac{k}{1+\varepsilon}}{\alpha^2\left(\frac{k}{1+\varepsilon}\right)^2} \leqslant \frac{1}{6} \Leftrightarrow 6 \cdot \frac{1+\varepsilon}{\varepsilon^2} \leqslant k.$$

Setting $k = 6 \times \left(\frac{1}{\varepsilon^2} + \frac{1}{\varepsilon}\right) = \Theta\left(\frac{1}{\varepsilon^2}\right)$ suffices to have

$$\Pr\left[\sum_{i=1}^{d} X_i > k\right] = \Pr[\hat{d} > (1+\varepsilon)d] \leqslant \frac{1}{6}. \tag{1}$$

On the other hand, by the same argument, we have $Y_i = \mathbb{1}_{\left\{h(i) \leqslant \frac{k}{(1-\varepsilon)d}\right\}}$

$$\begin{aligned}
\hat{d} < (1-\varepsilon)d \quad &\Leftrightarrow \quad \frac{k}{\max(z_1,\ldots,z_k)} < (1-\varepsilon)d \\
&\Leftrightarrow \quad \frac{k}{(1-\varepsilon)d} < \max(z_1,\ldots,z_k) \\
&\Leftrightarrow \quad \text{at most } k-1 \text{ (out of } d \text{ distinct points) of them have } h(i) \leqslant \frac{k}{(1-\varepsilon)d}. \\
&\Leftrightarrow \quad \sum_{i=1}^{d} Y_i < k.
\end{aligned}$$

Using same argument above, we compute

$$\mathbb{E}\left[\sum_{i=1}^{d} Y_i\right] = \frac{k}{(1-\varepsilon)d} \cdot d = \frac{k}{1-\varepsilon}.$$

[1]Ignoring the error incurred due to discretization from $[0,1]$ to $\left\{0, \frac{1}{N}, \frac{2}{N}, \ldots, 1\right\}$, which is at most $\frac{1}{N}$.

$$\mathrm{Var}\left[\sum_{i=1}^{d} Y_i\right] = \sum_{i=1}^{d} \mathrm{Var}[Y_i] \leqslant \frac{k}{1-\varepsilon}.$$

By Chebyshev's inequality,

$$\Pr\left[\sum_{i=1}^{d} Y_i < (1-\alpha)\frac{k}{1-\varepsilon}\right] \leqslant \frac{\mathrm{Var}\left[\sum_{i=1}^{d} Y_i\right]}{\left(\alpha\mathbb{E}\left[\sum_{i=1}^{d} Y_i\right]\right)^2}.$$

Setting $\alpha = \varepsilon$,

$$\Pr\left[\sum_{i=1}^{d} Y_i < k\right] \leqslant \frac{\frac{k}{1-\varepsilon}}{\left(\varepsilon\frac{k}{1-\varepsilon}\right)^2} = \frac{1-\varepsilon}{\varepsilon^2 k} \leqslant \frac{1}{6} \Leftrightarrow 6\left(\frac{1}{\varepsilon^2} - \frac{1}{\varepsilon}\right) \leqslant k$$

$$\Pr\left[\sum_{i=1}^{d} Y_i < k\right] = \Pr[\hat{d} < (1-\varepsilon)d] \leqslant \frac{1}{6}. \tag{2}$$

It suffices to have $k = \Theta\left(\frac{1}{\varepsilon^2}\right)$. By a union bound combining (1) and (2), we have that

$$\Pr[(1+\varepsilon)d > \hat{d} > (1-\varepsilon)d] \geqslant 1 - \frac{1}{3}.$$

b) This will depend on the value of $k = \Theta\left(1/\varepsilon^2\right)$. We need to store the hash function $O(\log N)$ and k hashed values ($k \log N$), all this multiplied by a $O\left(\log\frac{1}{\delta}\right)$ factor for the median trick.

$$O\left(\frac{\log N}{\varepsilon^2} \cdot \log(1/\delta)\right) = O\left(\frac{\log n}{\varepsilon^2} \cdot \log\frac{1}{\delta}\right).$$

Advanced

Problem 8. Given a stream σ of length m where each element a_i is a vector in $\{-1,1\}^d$, our goal is to estimate how large the mean vector

$$\bar{a} := \frac{1}{m}\sum_{i=1}^{m} a_i \in [-1,1]^d$$

is: that is, to obtain a multiplicative estimate of $\|\bar{a}\|_2$ within a factor 2, with probability at least 99%. Assuming that the algorithm does not "pay" the cost of storing the random bits it uses, describe an approach to do this with very small space complexity. What is the number of "free random bits" your algorithm uses?

Solution 8. Ignoring the costs of keeping the JL random matrix A in memory. We have by the JL lemma the guarantee that for any one fixed $x \in \mathbb{R}^d$

$$\Pr[|\|Ax\|_2^2 - \|x\|_2^2| > \varepsilon\|x\|_2^2] < \delta,$$

where $A \in \mathbb{R}^{k\times d}$ and $k = O\left(\varepsilon^{-2}\log\frac{1}{\delta}\right)$. Take $\varepsilon = \frac{1}{2}$ and $\delta = O(1)$. So we can simply draw one random A (which costs $O(d)$ random bits if done naively[2]) and use it to compute a running average, which just needs saving one real number and a counter for m.

[2]One can however do better as shown in `https://cseweb.ucsd.edu/~dakane/deranomizedJL.pdf`, but this result is well beyond the scope of this class.