CS 514: Advanced Algorithms II – Sublinear Algorithms **Rutgers: Fall 2021**

Lecture 1

September 7, 2021

Instructor: Sepehr Assadi *Scribe: Sepehr Assadi*

Topics of this Lecture

1 Introduction to Sublinear Algorithms

As (theoretical) computer scientists, we have long considered algorithms with linear time or space[1] as the golden standard of achievement. After all, it is hard to imagine doing much better as any algorithm for solving a nontrivial problem needs to at least read or store the entire input once, no?

Nevertheless, in this age of "big data", how can we process massive datasets that are so immense that even most basic linear time or space algorithms can become computationally prohibitive on these inputs? This is the focus of *sublinear algorithms*, namely, algorithms whose resource requirements (e.g. time or space) are substantially smaller than the size of the input that they operate on.

We will study various advanced algorithmic ideas through the lens of sublinear algorithms in this course. In particular, we consider two most canonical models of sublinear algorithms, namely, sublinear *time* algorithms and (sublinear *space*) streaming algorithms, and cover several key algorithmic techniques in these (and related) models, as well as discuss limitations inherent to computing with constrained resources.

[1]Linear here means asymptotically equal to the input size.

1.1 Introduction to Sublinear Time Algorithms: A Toy Example

Sublinear time algorithms are those algorithms that solve a given problem (exactly or approximately, and often with the help of randomization) by reading only a minuscule fraction of the input[2] and have time complexity which is sublinear in the input size. In the following, we give a (completely) toy example of a problem that admits a sublinear time algorithm.

Problem 1 (Finding an even number)**.** Given an array $A[1:n]$ consisting of a permutation of $\{1, \ldots, n\}$ for $n > 1$, output an index i such that $A[i]$ is an even number.

There is of course a trivial $O(n)$ time algorithm for this problem. Iterate over the array until you find an even number (which is bound to happen for $n > 1$). However, can we design a sublinear time algorithm for this problem?

- *No* if we want a *deterministic* algorithm that always output the correct answer and always run in $o(n)$ time. We will see how to prove this formally in a couple of lectures but for now, let us give an intuitive answer: if the algorithm is deterministic, it can become "unlucky" in the sense that the first $n/2$ numbers from the array that it reads are all odd numbers, so it needs to take $\Omega(n)$ time to find an even number. *Note:* This is *not* a formal argument and *not* a proof!
- *Yes* if we allow a *randomized* algorithm that always output the correct answer and *in expectation* runs in $O(1)$ time. The algorithm is simply as follows: pick a random number p from $\{1, \ldots, n\}$ uniformly at random (we assume this step can be done in $O(1)$ time); check if $A[p]$ is even or not; repeat until you find a p where $A[p]$ is even. Since the number of even numbers in $\{1, \ldots, n\}$ is almost a half (± 1 depending on whether n is odd or even), each choice of p finds an even number with probability almost a half, and hence in expectation we only need to repeat this process $O(1)$ time.

We just saw our first sublinear time algorithm in this course (albeit a straightforward one for a silly problem). Interestingly, even this simple example shows that randomization is almost (but not quite) always needed for obtaining a sublinear time algorithm. In the remainder of this course, we will examine various other sublinear time algorithms for problems in graph optimization, property testing, distribution testing, etc.

1.2 Introduction to Sublinear Space Streaming Algorithms: A Toy Example

Streaming algorithms are those algorithms that solve a given problem (exactly or approximately, and often with the help of randomization) by reading the input "on the fly" without storing it, namely, by processing the input in a streaming fashion and using a sublinear memory[3]. In the following, we give a (completely) toy example of a problem that admits a sublinear space streaming algorithm.

Problem 2 (Finding a missing number)**.** Given an array $A[1:n-1]$ presented one by one in a stream (i.e., $A[1]$ arrives first, then $A[2]$, then $A[3]$, and so on), consisting of a permutation of $\{1, \ldots, n\}$ *minus* one (unknown) missing number i, output the missing number i.

There is of course a trivial $O(n)$ space algorithm for this problem. While iterating over the stream, maintain an n-bit array $B[1:n]$ and mark $B[A[i]] = 1$ when visiting number $A[i]$ of the stream; then, output the unique j where $B[j] = 0$. However, can we design an $o(n)$ space streaming algorithm for this problem?

- *Yes* even if we want a *deterministic* algorithm that always output the correct answer and always uses $O(\log n)$ bits of space. The algorithm is simply as follows. Maintain a single number M which is

[2]Typically, one needs to assume *query access* to the input (say, given query access to the adjacency matrix of a graph) when working with sublinear time algorithms. We will elaborate more on this in the next lectures.

[3]We will formalize streaming model later in this course.

initially zero, and update $M \leftarrow M + A[i]$ after visiting the i-th number in the stream. At the end, output:

$$\frac{n \cdot (n+1)}{2} - M = \sum_{i=1}^{n} i - M = \sum_{i=1}^{n} i - \sum_{j=1}^{n-1} A[j] = \text{missing number}.$$

Since $M \leq n^2$, we can store M in $O(\log n)$ bits.

Exercise: Prove that $\Omega(\log n)$ bits are needed for any streaming algorithm for this problem (this is true for both randomized and deterministic algorithms but you may find it easier to prove it for deterministic algorithms first).

Exercise: Prove that $O(k^2 \cdot \log n)$ bits are sufficient for a generalization of the problem where exactly k elements from $\{1, \ldots, n\}$ are missing from the stream.

Exercise:[4] The above bound for the k missing element problem is *not* optimal. If you really like to challenge yourself, prove that $\Theta(k \cdot \log(n/k))$ bits are necessary and sufficient for this problem.

Again, we just saw our first streaming algorithm in this course. However, it is worth mentioning that even though this algorithm was able to find the correct answer deterministically, most (but not all) of the streaming algorithms we are going to see again require randomization and approximation.

1.3 Notions of Approximation and Randomization for Sublinear Algorithms

Conceptually, sublinear algorithms aim to *obtain something for almost nothing*: this means that they aim to provide "some information" about a problem at a minuscule cost of resources. We already addressed the second part, namely, the "almost nothing" part: the resource requirement of the algorithm should be (potentially much) less than the input size itself. As for the first part of "obtaining something", this means that these algorithms need to use both approximation and randomization.

Approximation. There are different notions of approximation one may need to consider for a problem but two general paradigms are the following:

- **"Classical" Approximation**: This means that the algorithm outputs an answer which is *close* to the actual answer to the problem. For instance, consider the problem of estimating the number of edges in a graph: we say that an algorithm for this problem outputs an α-approximation (for some $\alpha > 1$) if and only if the output $\widetilde{m}$ of the algorithm satisfies the following equation (m is the actual number of edges, i.e., the correct answer):

$$\frac{1}{\alpha} \cdot m \leq \widetilde{m} \leq \alpha \cdot m.$$

  We elaborate more on this from the next lecture.

- **Property Testing Approximation**: This is a notion of approximation for *decision* problems where the answer is boolean (and hence classical approximation has no real meaning). This means that the algorithm outputs an answer which is a correct answer not necessarily for the current input, but for some input which is *close* to the current one. For instance, consider the problem of deciding whether a graph is connected or not: we say that an algorithm for this problem is an ε-error property tester if and only if the algorithm outputs 'connected' on graphs which are actually connected and outputs 'disconnected' on the graphs that cannot be made connected even by changing ε fraction of their edges (the answer can be arbitrary for graphs that do not fit neither of these definitions). We will formalize this notion in a couple of lectures.

[4]This is not an easy question (specially if you do not have the background for it) but it is not impossibly hard.

Roughly speaking one can consider classical approximation as approximation in the output domain (the output can be changed slightly to make the answer correct) while property testing approximation is approximation in the input domain (the input can be changed slightly to make the answer correct). Each notion has its own use depending on the application and we will consider both throughout this course.

Randomization. We will also typically need randomization in our algorithms. Usually this means that:

- On *any* input, the answer is correct (or approximately correct according to definitions above) with large constant probability, say 9/10; and,
- On *any* input, the resource requirement of the algorithm is bounded (appropriately) with large constant probability, say 9/10.

In particular, we still analyze randomized algorithms against *worst case inputs.*

1.4 Impossibility Results for Sublinear Algorithms

An interesting aspect of studying sublinear algorithms are *lower bounds* or *impossibility results*: these are the type of arguments that lower bound the resource requirement of a certain problem in a particular model (e.g., solving missing number problem requires $\Omega(\log n)$ bits in the streaming model) or show that a problem does not admit any sublinear algorithm. You might be familiar with such reasoning through notion of NP-hardness and other complexity classes for classical algorithms. However, the types of lower bounds (or rather impossibility results) proven for sublinear algorithms tend to be very different from NP-hardness results, primarily because for sublinear algorithms, we can prove *unconditional* lower bounds (unlike lower bounds for NP-hard problems that are conditioned on the hypothesis P $\neq$ NP or even much stronger hypotheses). Moreover, lower bounds for sublinear algorithms are primarily *information-theoretic* rather than being *computational* (we will get to this as well in a couple of lectures).

In this course, we will study query complexity and communication complexity as two primary ways of proving lower bounds for sublinear time and streaming algorithms, respectively.

2 Background: Probabilistic Analysis

The only background (beside basic familiarity with TCS concepts at the undergraduate level) that we need for this course is probabilistic analysis of algorithms and in particular concentration results. The rest of this lecture is dedicated to this topic. We will study these topics through the lens of another toy problem:

Problem 3 (Balls-in-Bins)**.** Consider the following probabilistic experiment: We are give a collection of n balls and n bins. We throw each ball *independently* to one of the bins chosen *uniformly at random.* We define *load* of a bin as the number of balls thrown to that bin in this experiment. What is the asymptotically maximum load of a bin in this process with high probability[5]?

More precisely, our goal is to find the smallest possible upper $T(n)$ (as a function of n) such that with probability at least, say, $1 - 1/n$, the load of *every* bin in this process is $O(T(n))$.

Exercise: For some students, depending on their background, this problem can be straightforward. If this is the case for you, think about the following problem instead: suppose for each ball, we sample *two* bins uniformly at random and then place the ball in the one with the smaller load (breaking the ties consistently, say, with the ID of the bins). Find the tightest bound you can on the maximum load of a bin with high probability in this alternate process (this bound is surprisingly much smaller than the original problem!).

Before getting to answer this question, let us fix a single bin, say bin 1, and simply consider the load on this single bin. Thus we define the following simpler variant of the problem:

[5]With high probability is a term used typically to refer to the probability of $1-1/\text{poly}(n)$, for some arbitrary large polynomial in n where n denotes the input size – while this may seem imprecise and vague at this point, we will see enough example of this in this course for you to get familiar with this concept.

Simpler problem: In the balls in bins experiment, what is an upper bound $F(n)$ such that with probability at least $1 - 1/n$, the load of the *first bin* in this process is $O(F(n))$?[6]

For the rest of this lecture, we focus on answering this simpler problem and then we will see how the answer to this problem (almost immediately) solves the original problem as well.

Solution: Define $L \in \{0, \ldots, n\}$ as the random variable denoting the load of the first bin in this process. Our goal is thus to find a function $F(n)$ such that,

$$\Pr(L \geq F(n)) \leq \frac{1}{n}. \tag{1}$$

As is typical in probabilistic analysis of algorithms, this is done in two steps:

1. Bounding the expected value of L, namely, $\mathbb{E}[L]$;
2. Bounding the probability that L is "very different" from its expected value[7].

Step 1 (Bounding $\mathbb{E}[L]$): What is the expected number of balls in bin one in this process? Let us define n random variables $B_1, \ldots, B_n$ where $B_i \in \{0, 1\}$ and $B_i = 1$ if and only if the i-th ball is thrown to bin one[8]. Then, $L = B_1 + B_2 + \ldots + B_n$ and so $\mathbb{E}[L] = \mathbb{E}[B_1 + \ldots + B_n]$. We are going to use the following extremely important property of expectation, the so-called *linearity of expectation*:

Fact 1 (**Linearity of Expectation**). *For any two random variables X, Y, $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$.*

Remark. Linearity of expectation ***always*** hold regardless of whether X, Y are independent or not; this is perhaps the single most important property of expectation which makes analyzing it much simpler than working with probabilities directly. In a nutshell, linearity of expectation is one of your strongest allies in analyzing any randomized process!

Using linearity of expectation, we can thus write,

$$\begin{aligned} \mathbb{E}[L] &= \mathbb{E}\left[\sum_{i=1}^{n} B_i\right] = \sum_{i=1}^{n} \mathbb{E}[B_i] && \text{(linearity of expectation)} \\ &= \sum_{i=1}^{n} \Pr(B_i = 1) = \sum_{i=1}^{n} \frac{1}{n} = 1. && \text{(each ball is "choosing" bin one with probability } 1/n\text{)} \end{aligned}$$

We are already done with step 1 of the argument. But bounding expectation is never enough; we should also make sure that our random variable does not behave very differently from its expectation.

We now get to the second and main part of the argument. We need to "translate" bounds on expectation to bounds on the probability. I.e., we want to know the answer to the following question: what is the probability that L is "very far" from its expectation? This is the focus of *concentration results.* We are going to consider three most basic concentration results that are used in the analysis of algorithms (and almost anywhere else that one uses randomization) and use them to answer this question.

[6]It is worth emphasizing again: the only difference between this problem and the original one is that we are now interested in only bounding the load of a single fixed bin, instead of every bin simultaneously.

[7]After all, we expect a random variable to behave as its expected value – the goal of this step is to transform this qualitative statement into a quantitative one.

[8]Such 0/1-random variables are called *indicator random variables* because they indicate (by getting value of one) when a particular event happens and otherwise are zero. Indicator random variables are used extensively in analysis of algorithms (and elsewhere).

2.1 Concentration Results

Roughly speaking, concentration results bound the *deviation* probability of a random variable from its expected value.

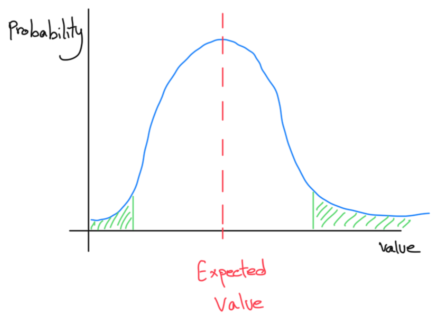

Figure 1: Concentration results bound the total probability mass of a distribution that are "far" from mean value of that distribution (red shaded part). The more information one has about the distribution, the stronger type of concentration one can prove about the distribution.

2.1.1 Markov Bound

The simplest and most basic variant of concentration results is *Markov bound* or *Markov inequality*:

Proposition 2 (Markov Bound). *For a* non-negative *random variable X and $t > 0$,*

$$\Pr(X \geq t \cdot \mathbb{E}[X]) \leq \frac{1}{t}.$$

Proof. Let $\mu := \mathbb{E}[X]$. We can use law of total conditional probabilities to have:

$$\begin{aligned}\mathbb{E}[X] &= \mathbb{E}[X \mid X \geq t \cdot \mu] \cdot \Pr(X \geq t \cdot \mu) + \mathbb{E}[X \mid X < t \cdot \mu] \cdot \Pr(X < t \cdot \mu) \\ &\geq t \cdot \mu \cdot \Pr(X \geq t \cdot \mu) + 0. \\ &\text{(the first term since we conditioned on } X \geq t \cdot \mu \text{ and the second term since } X \text{ is non-negative)}\end{aligned}$$

Thus, $\Pr(X \geq t \cdot \mathbb{E}[X]) = \Pr(X \geq t \cdot \mu) \leq 1/t$, otherwise the RHS above will be larger than the LHS. □

Markov bound only bounds the *upper tail* of the distribution[9]: the probability that a random variable takes value t times larger than its expectation is at most $1/t$. This is a basic but extremely useful property that is used extensively. One can alternatively state the Markov bound as follows.

Corollary 3. *For a* non-negative *random variable X and $b > 0$,*

$$\Pr(X \geq b) \leq \frac{\mathbb{E}[X]}{b}.$$

Proof. The proof is by simply picking $t = b/\mathbb{E}[X]$ in Proposition 2. □

[9]Although one can use it to bound the lower tail in special cases as well, but the bounds there are generally very weak.

Markov bound for balls-and-bins experiment? Let us now apply Markov bound to our balls in bins experiment to bound $F(n)$. By Markov bound, $\Pr\left(L \geq n \cdot \mathbb{E}\left[L\right]\right) \leq {}^{1}/_{n}$; hence, applying Markov bound only allows us to bound the load of *a single* bin by n, which is completely trivial!

This is not surprising since we did not use any property of the random variable L beside the fact that it was non-negative. In particular, the same analysis applies to the case where instead of sending the balls *independently*, we could pick a single bin uniformly at random and send all balls into that bin. In that case Markov bound would clearly give the tightest possible bound on $F(n)$. As such, to get better bounds for our random variable L, we should use our extra knowledge of L.

Remark. Even though Markov bound may sound almost trivial (and it is indeed straightforward), it is the basis for proving all other concentration inequalities that we use in this course; moreover, Markov bound is used one way or another in analysis of almost every algorithm.

2.1.2 Chebyshev's Inequality

We now consider our second concentration inequality: *Chebyshev inequality.* Unlike Markov bound that only required a knowledge of the expected value of the random variable to bound its deviation probability, Chebyshev's inequality applies to the settings in which we could additionally bound the *variance* of the random variable as well.[10]

Recall that for a random variable X, *variance* of X is:

$$\mathrm{Var}\left[X\right] := \mathbb{E}\left[(X - \mathbb{E}\left[X\right])^2\right] = \mathbb{E}\left[X^2\right] - (\mathbb{E}\left[X\right])^2.$$

Chebyshev inequality allows us to bound deviation of a random variable based on its variance.

Proposition 4 (Chebyshev's Inequality). *For any random variable X and $t > 0$,*

$$\Pr\left(\left|X - \mathbb{E}\left[X\right]\right| \geq t \cdot \mathbb{E}\left[X\right]\right) \leq \frac{\mathrm{Var}\left[X\right]}{\mathbb{E}\left[X\right]^2 \cdot t^2}.$$

Proof. Define a new random variable $Y := (X - \mathbb{E}\left[X\right])^2$. Clearly, Y is non-negative. Moreover, $\left|X - \mathbb{E}\left[X\right]\right| \geq t \cdot \mathbb{E}\left[X\right]$ if and only if $Y = (X - \mathbb{E}\left[X\right])^2 \geq t^2 \cdot \mathbb{E}\left[X\right]^2$. Hence,

$$\begin{aligned}\Pr\left(\left|X - \mathbb{E}\left[X\right]\right| \geq t \cdot \mathbb{E}\left[X\right]\right) = \Pr\left(Y \geq t^2 \cdot \mathbb{E}\left[X\right]^2\right) &\leq \frac{\mathbb{E}\left[Y\right]}{\mathbb{E}\left[X\right]^2 \cdot t^2} && \text{(by Corollary 3)} \\ &= \frac{\mathrm{Var}\left[X\right]}{\mathbb{E}\left[X\right]^2 \cdot t^2}. && \text{(as } \mathbb{E}\left[Y\right] = \mathbb{E}\left[(X - \mathbb{E}\left[X\right])^2\right] = \mathrm{Var}\left[X\right] \text{ by definition)}\end{aligned}$$

□

A useful variant of Chebyshev's inequality is the following.

Corollary 5. *For a random variable X and $b > 0$,*

$$\Pr\left(\left|X - \mathbb{E}\left[X\right]\right| \geq b\right) \leq \frac{\mathrm{Var}\left[X\right]}{b^2}.$$

Proof. The proof is by simply picking $t = {}^{b}/_{\mathbb{E}[X]}$ in Proposition 4. □

[10] As we showed earlier, Markov bound *can* be tight for certain random variables; thus, naturally whenever we need a stronger bound we should show that our variable satisfies additional guarantees that what is only required by Markov bound.

Chebyshev's inequality for balls-in-bins experiment? We now apply Chebyshev's inequality to the random variable $L = B_1 + B_2 + \ldots + B_n$ where B_i is the indicator random variable for i-th ball to be sent to the first bin. In general, variance is not a *linear* function (unlike expectation). However, for independent random variables, variance of the sum is equal to the sum of variances. I.e.,

Proposition 6. *For independent random variables X, Y,* $\text{Var}[X+Y] = \text{Var}[X] + \text{Var}[Y]$.

Proof. We have,

$$\begin{aligned}
\text{Var}[X+Y] &= \mathbb{E}\left[(X+Y)^2\right] - (\mathbb{E}[X+Y])^2 = \mathbb{E}\left[X^2+Y^2+2XY\right] - (\mathbb{E}[X]^2 + \mathbb{E}[Y]^2 + 2\,\mathbb{E}[X]\,\mathbb{E}[Y]) \\
&= \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 + \mathbb{E}\left[Y^2\right] - \mathbb{E}[Y]^2 + 2(\mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y]) \quad \text{(by linearity of expectation)} \\
&= \text{Var}[X] + \text{Var}[Y] + 0. \qquad (\text{as } \mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y] \text{ for } \textit{independent} \text{ random variables } X, Y)
\end{aligned}$$

□

Remark. To emphasize again, *unlike expectation, variance is not a linear function* in general (we do not have linearity of variance). But, variance is linear when working with ***independent*** random variables.

Applying Proposition 6 to the random variable $L = B_1 + \ldots + B_n$ and using the fact that B_i's are independent, we have,

$$\begin{aligned}
\text{Var}[L] &= \sum_{i=1}^{n} \text{Var}[B_i] = \sum_{i=1}^{n} \mathbb{E}\left[B_i^2\right] - \mathbb{E}[B_i]^2 \leq \sum_{i=1}^{n} \mathbb{E}\left[B_i^2\right] \\
&= \sum_{i=1}^{n} \mathbb{E}[B_i] = n \cdot \frac{1}{n} = 1,
\end{aligned}$$

where the first equality in the second line is because B_i is an indicator random variable and hence $B_i^2 = B_i$ (as $1^2 = 1$ and $0^2 = 0$).

We can now apply Chebyshev's inequality and obtain that,

$$\Pr\left(|L - \mathbb{E}[L]| \geq \sqrt{n} \cdot \mathbb{E}[L]\right) \leq \frac{\text{Var}[L]}{\mathbb{E}[L]^2 \cdot n} \leq \frac{1}{n}.$$

Hence, Chebyshev's inequality allows us to bound the load of *a single* bin by $\sqrt{n}$ with probability $1 - 1/n$, i.e., set $F(n) = \sqrt{n}$. Unlike the conclusion of Markov bound, this is already non-trivial, albeit still very far from the right answer.

2.1.3 Chernoff Bound

We now consider our third and strongest concentration inequality, namely, the *Chernoff bound.* To be able to use Chernoff bound for bounding deviation of a random variable X, we need a much stronger knowledge than variance in Chebyshev's inequality and expectation in Markov bound; we now need to know that the random variable X is a *sum of bounded-value independent random variables.* Formally,

Proposition 7 (Chernoff Bound). *Suppose $X_1, \ldots, X_n$ are* independent *random variables in $[0,1]$ and $X = \sum_i X_i$. Then, for any $t \geq 1$,*

$$\Pr\left(|X - \mathbb{E}[X]| \geq t \cdot \mathbb{E}[X]\right) \leq 2 \cdot \exp\left(-\frac{t \cdot \mathbb{E}[X]}{3}\right).$$

Moreover, for any $\varepsilon \in (0,1]$,

$$\Pr\left(|X - \mathbb{E}[X]| \geq \varepsilon \cdot \mathbb{E}[X]\right) \leq 2 \cdot \exp\left(-\frac{\varepsilon^2 \cdot \mathbb{E}[X]}{3}\right).$$

Proof of a weaker form. We are *not* going to give the entire proof of the Chernoff bound as it is somewhat tedious. However, to provide enough intuition, we will prove a simpler variant of Chernoff (which is actually sufficient for our balls in bins argument and many other settings as well). In particular, we are going to make the following simplifying assumption.

Simplifying assumption: Let us assume that each X_i is a Bernoulli random variable[11] with mean p_i (instead of arbitrary random variable in $[0,1]$). For simplicity, we are also going to only prove the *upper tail* of the deviation bound instead of both tails (but the lower tail can be proven symmetrically).

Fix $\alpha > 0$. Note that $X \geq (1+\varepsilon)\,\mathbb{E}[X]$ if and only if $\exp(\alpha \cdot X) \geq \exp(\alpha \cdot (1+\varepsilon) \cdot \mathbb{E}[X])$. Define a random variable $Y = \exp(\alpha \cdot X)$. Using Markov bound on random variable Y, we have,

$$\Pr(X \geq (1+\varepsilon)\cdot\mathbb{E}[X]) = \Pr(Y \geq \exp(\alpha\cdot(1+\varepsilon)\cdot\mathbb{E}[X])) \leq \frac{\mathbb{E}[Y]}{\exp(\alpha\cdot(1+\varepsilon)\cdot\mathbb{E}[X])} \tag{2}$$

As such, to bound the probability of deviation of X from $\mathbb{E}[X]$, we only need to bound $\mathbb{E}[Y]$ and then we can apply Eq (2). We can now upper bound $\mathbb{E}[Y]$ as follows:

$$\begin{aligned}
\mathbb{E}[Y] = \mathbb{E}[\exp(\alpha\cdot X)] &= \mathbb{E}\left[\exp\left(\alpha\cdot\sum_{i=1}^n X_i\right)\right] \\
&= \mathbb{E}\left[\prod_{i=1}^n \exp(\alpha\cdot X_i)\right] \\
&= \prod_{i=1}^n \mathbb{E}[\exp(\alpha\cdot X_i)] && \text{(for \underline{independent} random variables } A,B\text{: } \mathbb{E}[AB]=\mathbb{E}[A]\cdot\mathbb{E}[B]) \\
&= \prod_{i=1}^n (1-p_i+p_i\cdot e^\alpha) && \text{(by the assumption that } X_i \text{ is Bernoulli with mean } p_i) \\
&= \prod_{i=1}^n (1+p_i\cdot(e^\alpha-1)) \\
&\leq \prod_{i=1}^n \exp(p_i\cdot(e^\alpha-1)) && (1+x\leq e^x \text{ for all } x) \\
&= \exp\left((e^\alpha-1)\cdot\sum_{i=1}^n p_i\right) \\
&= \exp((e^\alpha-1)\cdot\mathbb{E}[X]).
\end{aligned}$$

Let us now set $\alpha = \ln(1+\varepsilon)$ and use Eq (2) to obtain that:

$$\begin{aligned}
\Pr(X\geq(1+\varepsilon)\cdot\mathbb{E}[X]) &\leq \exp((e^\alpha-1)\cdot\mathbb{E}[X]-\alpha\cdot(1+\varepsilon)\cdot\mathbb{E}[X]) \\
&= \exp(\varepsilon\cdot\mathbb{E}[X]-(1+\varepsilon)\cdot\ln(1+\varepsilon)\cdot\mathbb{E}[X]) \\
&= \left(\frac{e^\varepsilon}{(1+\varepsilon)^{(1+\varepsilon)}}\right)^{\mathbb{E}[X]}.
\end{aligned} \tag{3}$$

The bound above is already a ***very strong form*** of Chernoff bound (for Bernoulli random variables) – note also that in this equation, we do *not* need ε to be in $(0,1)$. We can simplify this bound to get the bounds we want in the statement of the Proposition 7 (this will weaker the bound slightly; very rarely this weakening can be problematic and we may need to use the stronger bound above directly).

We are going to use the following inequality (the proof is omitted) to simplify Eq (3): For any $x > 0$,

$$1+x \geq \frac{e^x}{1+x/2}.$$

[11]Recall that a Bernoulli random variable Z with mean p gets value 1 w.p. p and 0 w.p. $1-p$.

By applying this, we have (the proof is again omitted),

$$\left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{(1+\varepsilon)}}\right) = \exp\left(\varepsilon - (1+\varepsilon)\cdot\ln(1+\varepsilon)\right) \leq -\frac{\varepsilon^2}{2+\varepsilon}.$$

And thus by Eq (3),

$$\Pr\left(X \geq (1+\varepsilon)\cdot\mathbb{E}[X]\right) \leq \exp\left(-\frac{\varepsilon^2}{2+\varepsilon}\cdot\mathbb{E}[X]\right).$$

Both bounds (the upper tail) in the proposition statement now follows from the above by considering $\varepsilon \geq 1$ and $\varepsilon < 1$ cases separately. □

Chernoff bound for balls-in-bins experiment? Finally, let us apply Chernoff bound to the random variable $L = B_1 + \ldots + B_n$ to obtain a better bound for $F(n)$ in our balls in bins experiment. Since B_i's are *independent* 0/1-random variables, by Chernoff bound (first equation of Proposition 7),

$$\Pr\left(|L - \mathbb{E}[L]| \geq 6\ln n\right) \leq 2\cdot\exp\left(-2\ln n\right) \leq \frac{2}{n^2}.$$

Hence, Chernoff bound allows us to bound $F(n) \leq 6\ln n + 1$ with probability $1 - 2/n^2$ already. We can also change the probability to any $1 - 1/n^c$ for any constant c by changing the leading constant behind $\ln n$ term.

2.2 Wrapping Up: The Union Bound

Recall that our goal is to bound $T(n)$, namely, the maximum load of every ball – in the above part however, we bound the load of any single (fixed) ball. We now extend the argument to answer the former (and main) question very easily. This is basically by using the following very simple fact about probabilities, referred to as the *union bound.*

Fact 8. *For any two events $\mathcal{E}_1, \mathcal{E}_2$, we have* $\Pr(\mathcal{E}_1 \cup \mathcal{E}_2) \leq \Pr(\mathcal{E}_1) + \Pr(\mathcal{E}_2)$.

For analyzing the function $T(n)$ in the balls and bins argument, we can simply set $T(n) = 12\ln n$ and use the argument from the previous section together with a union bound to have that,

$$\begin{aligned}\Pr\left(\text{there exists a bin with load more than } T(n) = 12\ln n\right) &\leq \sum_{i=1}^{n} \Pr\left(\text{bin } i \text{ has load more than } 12\ln n\right) \\ &\leq n\cdot\frac{2}{n^4} \leq \frac{1}{n^2}. \qquad \text{(for } n > 1\text{)}\end{aligned}$$

As such, we proved that the maximum load of any ball is $O(\log n)$ in this experiment with high probability.

Note: While $O(\log n)$ is very close to the "right" answer (asymptotically) for this problem, it is still *not* quite the tightest bound. We revisit this in the first problem set.

Remark. You may wonder why we consider simple tools like linearity of expectation or union bound as extremely important. The main reason is that these tools allow us to reason about a complex random variable – a one consisting of different components or and other random variables – by "breaking" them down to simpler random variables and analyzing these simpler parts in isolation instead, a task which is much simpler than analyzing the probability distribution of the original complex random variable directly (on this front, linearity of expectation is particularly appealing because it is an "equality" and thus this process will actually find the tight answer to our problem).