CS166 Problem Set 0: Concept Refresher

[Luis A. Perez]

Section One: Mathematical Prerequisites

Problem One: Fibonacci Fun! (3 Points)

The Fibonacci numbers are a famous sequence defined as

$$F_0 = 0 \qquad F_1 = 1 \qquad F_{n+2} = F_n + F_{n+1}$$

For example, the first few terms of the Fibonacci sequence are

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

There's a close connection between the Fibonacci numbers and the quantity $\varphi = \frac{1+\sqrt{5}}{2}$, the ***golden ratio***. In case you're wondering where that number comes from, the golden ratio is the positive root of the quadratic equation $x^2 = 1 + x$.

We'd like you to prove some results about the Fibonacci numbers. In what follows, please do not use any properties of Fibonacci numbers other than what's given in the definition above. The purpose of this problem is to make sure you're comfortable reasoning about terms from first principles.

i. Using the formal definition of big-O notation, prove that $F_n = O(\varphi^n)$. To do so, find explicit choices of the constants c and n_0 for the definition of big-O notation, then use induction to prove that those choices are correct.

   Our proofwriting style expectations are along the lines of what you'd see in CS161. Write in complete sentences rather than bullet points, use mathematical notation when appropriate but not as a stand-in for plain English, etc. Remember that an actual person is going to be reading your proof, so be nice to them by writing a lucid, clear argument that respects the intelligence of the reader but doesn't ask them to do the heavy lifting for you. ☺

   Solution: To formally prove that $F_n = O(\varphi^n)$, we must show that $\exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R}$ such that $\forall n \in \mathbb{N}$, if $n \geq n_0$, then $F_n \leq c\varphi^n$. We will do this by strong induction on n using the constants $n_0 = 0$ and $c = 1$. We arrived at these constants by attempting the proof and working out the values. We begin with two base cases: $n = 0$ and $n = 1$. We have:

$$F_0 = 1 \leq \varphi^0 = 1$$
$$F_1 = 1 \leq \varphi^1 = \frac{1+\sqrt{5}}{2} \approxeq 1.618$$

We now state formally the strong inductive hypothesis. Let us assume that for all $n \leq k$, the following inequality holds (for an example, our base-cases proves this for $k = 1$):

$$F_n \leq c\varphi^n = \varphi^n$$

We now seek to demonstrate that given the above, for $n = k + 1$, the following also holds:

$$F_{k+1} \leq \varphi^{k+1}$$

As we'll see below, this eventually boils down to solving the following inequality for c:

$$c\varphi^{k+1} \leq c\varphi^k + c\varphi^{k-1}$$
$$\implies c\varphi^2 \leq c[\varphi + 1]$$

which turns out to hold true for all values of c, and in particular, for $c = 1$, since in that case we simply have the statement $\varphi^2 \leq \varphi + 1$. As we see above, any c satisfying the restriction of big-O as well as large-enough to satisfy our base-cases will work. Specifically, we must have $c \in [1, \infty)$ for $n_0 = 0$. We pick $c = 1$ for simplicity.

$$\begin{aligned}
F_{k+1} &= F_k + F_{k-1} && \text{(By definition of } F_n\text{)} \\
&\leq \varphi^k + \varphi^{k-1} && \text{(By strong induction)} \\
&= \varphi^{k-1}[\varphi + 1] && \text{(Factoring out } \varphi^{k-1}\text{)} \\
&= \varphi^{k-1}\varphi^2 && (\varphi + 1 = \varphi^2) \\
&= \varphi^{k+1}
\end{aligned}$$

Therefore, given our strong inductive hypothesis, we have show that $F_{k+1} \leq \varphi^{k+1}$. Putting it all together, the above shows that for all $n \geq n_0$, $F_n \leq c\varphi^n$ where $n_0 = 0$ and $c = 1$. We conclude that $F_n = O(\varphi^n)$.

ii. Along the lines of part (i) of this problem, using the formal definition of big-Ω notation, prove that $F_n = \Omega(\varphi^n)$.

Solution: To formally prove that $F_n = \Omega(\varphi^n)$, we must show that $\exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R}$ such that $\forall n \in \mathbb{N}$, if $n \geq n_0$, then $F_n \geq c\varphi^n$. We will do this by strong induction on n using the constants $n_0 = 0$ and $c = \frac{1}{\varphi}$. We arrived at these constants by attempting the proof and working out the values. We begin with two base cases: $n = 0$ and $n = 1$. We have:

$$F_0 = 1 \geq \frac{1}{\varphi}\varphi^0 = \frac{1}{\varphi} \approxeq 0.618$$
$$F_1 = 1 \geq \frac{1}{\varphi}\varphi^1 = 1$$

We now state formally the strong inductive hypothesis. Let us assume that for all $n \leq k$, the following inequality holds (for an example, our base-cases proves this for $k = 1$):

$$F_n \geq c\varphi^n = \frac{1}{\varphi}\varphi^n$$

We now seek to demonstrate that given the above, for $n = k + 1$, the following also holds:

$$F_{k+1} \geq \frac{1}{\varphi}\varphi^{k+1}$$

As we'll see below, this eventually boils down to solving the following inequality for c:

$$\begin{aligned} c\varphi^{k+1} &\geq c\varphi^k + c\varphi^{k-1} \\ \implies c\varphi^2 &\geq c[\varphi + 1] \end{aligned}$$

which turns out to hold true for all values of c, and in particular, for $c = \frac{1}{\varphi}$. As we see above, any c satisfying the restriction of big-Ω as well as small-enough to satisfy our base-cases will work. Specifically, we must have $c \in (0, \frac{1}{\varphi}]$ for $n_0 = 0$. We pick $c = \frac{1}{\varphi}$ for simplicity.

$$\begin{aligned} F_{k+1} &= F_k + F_{k-1} && \text{(By definition of } F_n\text{)} \\ &\geq \frac{1}{\varphi}\varphi^k + \frac{1}{\varphi}\varphi^{k-1} && \text{(By strong induction)} \\ &= \frac{1}{\varphi}\varphi^{k-1}[\varphi + 1] && \text{(Factoring out } \varphi^{k-1}\text{)} \\ &= \frac{1}{\varphi}\varphi^{k-1}\varphi^2 && (\varphi + 1 = \varphi^2) \\ &= \frac{1}{\varphi}\varphi^{k+1} \end{aligned}$$

Therefore, given our strong inductive hypothesis, we have show that $F_{k+1} \geq \frac{1}{\varphi}\varphi^{k+1}$. Putting it all together, the above shows that for all $n \geq n_0$, $F_n \geq c\varphi^n$ where $n_0 = 0$ and $c = \frac{1}{\varphi}$. We conclude that $F_n = \Omega(\varphi^n)$.

You've just proved that $F_n = \Theta(\varphi) = \Theta(\varphi^n)$, which is not immediately obvious! Fibonacci numbers show up in lots of algorithms and data structures, and what you've just proved will definitely make an appearance later this quarter.

Problem Two: Probability and Concentration Inequalities (4 Points)

The analysis of randomized data structures sometimes involves working with sums of random variables. Our goal will often be to get a tight bound on those sums, usually to show that some runtime is likely to be low or that some estimate is likely to be good. If we only know two pieces of information about those random variables (what their expected values are and that they're nonnegative), we can get some information about how their sums behave.

i. Let $X_1, X_2, \ldots, X_n$ be a collection of n nonnegative random variables such that $\mathbb{E}[X_i] = 1$ for each variable X_i. (Note that these random variables might not be independent of one another.) Prove that $\Pr\left[\sum_{i=1}^n X_i \geq 2n\right] \leq \frac{1}{2}$. You may want to use Markov's inequality.

Solution: Let us begin by considering the random variable Y defined as $Y = \sum_{i=1}^n X_i$ (eg, the sum of our collection of X_i). We compute $\mathbb{E}[Y]$ first.

$$\begin{aligned}
\mathbb{E}[Y] &= \mathbb{E}\left[\sum_{i=1}^n X_i\right] && \text{(Definition of } Y\text{)} \\
&= \sum_{i=1}^n \mathbb{E}[X_i] && \text{(Linearity of expectation, which holds for any r.v)} \\
&= \sum_{i=1}^n 1 && (\mathbb{E}[X_i] = 1 \text{ as given in the problem)} \\
&= n
\end{aligned}$$

We now note that Y is a non-negative random variable (it is the sum of non-negative random variables) and has a finite-expected value. We can apply Markov's inequality, and do so with $c = 2$ to have the following:

$$\begin{aligned}
\Pr\left[Y \geq 2\mathbb{E}[Y]\right] &\geq \frac{1}{2} && \text{(Markov Inequality applied to } Y \text{ with } c = 2\text{)} \\
\iff \Pr\left[\sum_{i=1}^n X_i \geq 2n\right] &\leq \frac{1}{2} && \text{(Definition of } Y \text{ and results from above)}
\end{aligned}$$

The above proves the statement we wanted to prove.

Sometimes you'll find that the sort of bound you get from an analysis like part (i) isn't strong enough to prove what you need to prove. In those cases, you might want to start looking more at the spread of each individual random variable. If, for example, you know the variances of those variables are small, you might be able to get a tighter bound.

ii. Let $X_1, X_2, \ldots, X_n$ be a collection of n nonnegative random variables. As in part (i), you know that $\mathbb{E}[X_i] = 1$ for each variable X_i. But now suppose you know two other facts. First, you know that each variable has unit variance (the fancy way of saying $\mathrm{Var}[X_i] = 1$ for each variable X_i). Second, while you don't know for certain whether these variables are independent of one another, you know that they're ***pairwise uncorrelated***. That is, you know that X_i and X_j are uncorrelated random variables for any $i \neq j$. Under these assumptions, prove that $\Pr\left[\sum_{i=1}^n X_i \geq 2n\right] \leq \frac{1}{n}$. You may want to use Chebyshev's inequality.

Solution: We follow a similar approach to the previous part, with the additional assumption that $n > 0$ (this is not given in the problem statement, but without this restriction, the given inequality is undefined). In particular, we again define the random variable $Y = \sum_{i=1}^n X_i$ which is a non-negative random variable. From the part above, we already know that $\mathbb{E}[Y] = n$. Since we know $\text{Var}[X_i] = 1$ and that the variables are pairwise uncorrelated, we begin by first using this information to find $\text{Var}[Y]$.

$$\begin{aligned}
\text{Var}[Y] &= \text{Var}\left[\sum_{i=1}^n X_i\right] \\
&= \sum_{i=1}^n \text{Var}[X_i] + \sum_{i \neq j} \text{Cov}[X_i, X_j] \\
&\qquad \text{(Properties of variance of any set of random variables)} \\
&= \sum_{i=1}^n \text{Var}[X_i] \\
&\qquad \text{(Variables are pair-wise uncorrelated, so Cov}[X_i, X_j] = 0 \text{ for } i \neq j) \\
&= n && (\text{Var}[X_i] = 1)
\end{aligned}$$

We now have enough information to prove the given statement. We begin by manipulating it a little bit.

$$\Pr\left[\sum_{i=1}^n X_i \geq 2n\right] = \Pr\left[\sum_{i=1}^n X_i - n \geq n\right]$$

From here, we note that we can apply an upper bound. Let A be the event where $\sum_{i=1}^n X_i - n \geq n$ and let B be the event where $|\sum_{i=1}^n X_i - n| \geq n$. Then note that $A \implies B$, since we know that $n > 0$. As such, If A occurs, then we know that $\sum_{i=1}^n X_i - n$ is positive, which implies B occurs. This means that $\Pr[B] \geq \Pr[A]$. Fully written out, we know:

$$\Pr\left[\sum_{i=1}^n X_i - n \geq n\right] \leq \Pr\left[|\sum_{i=1}^n X_i - n| \geq n\right]$$

We can now make some subtitutions to make this resemble Chebyshev's inequality and thereby make use of that known result. In particular, recall that $Y = \sum_{i=1}^n X_i$,

$\mathbb{E}[Y] = n$, and $\text{Var}[Y] = n$. We therefore have:

$$\Pr\left[|\sum_{i=1}^{n} X_i - n| \geq n\right] = \Pr\left[|Y - \mathbb{E}[Y]| \geq \sqrt{n}\sqrt{\text{Var}[Y]}\right]$$
$$\leq \frac{1}{n}$$
$$\text{(By Chebyshev's inequality with } c = \sqrt{n} > 0\text{)}$$

Putting it all together, for non-negative, pair-wise uncorrelated X_i with unit variance we have:

$$\Pr\left[\sum_{i=1}^{n} X_i \geq 2n\right] \leq \frac{1}{n}$$

The analysis in part (ii) only works if the variables are pairwise uncorrelated.

iii. Pick a natural number $n > 0$ and define a collection of random variables $X_1, X_2, \ldots, X_n$ such that

- each X_i is nonnegative,
- $\mathbb{E}[X_i] = 1$ for each variable X_i,
- $\text{Var}[X_i] = 1$ for each variable X_i, but
- $\Pr\left[\sum_{i=1}^{n} X_i \geq 2n\right] > \frac{1}{n}$.

Once you've done this, go back to your proof from part (ii) and make sure you can point out the specific spot where the math breaks down once you remove the requirement that the Xi's be pair- wise uncorrelated.

Solution: We take $n = 3$. Let X_1, X_2, X_3 be the random variables taking on two values, 0 and 2 with equal probability. It is straight-forward to confirm that each X_i is non-negative, $\mathbb{E}[X_i] = 1$ (the average of the two values), and that $\text{Var}[X_i] = \mathbb{E}[(X - \mathbb{E}[X_i])^2] = 1$ (distance from the expected value is always 1). However, the X_i such that in all realization, we must have that $X_i = X_j$ for all i, j. We therefore have:

$$\Pr\left[\sum_{i=1}^{3} X_i \geq (2)(3)\right] = \frac{1}{2} \geq \frac{1}{3}$$

This is because the sum of the variables can be either $0 + 0 + 0$ or $2 + 2 + 2 = 6$, each even with equal probability. Note that this easily extends for arbitrary values of n. This, however, does not contradict our proof from above since this break the assumption we made where $\text{Cov}[X_i, X_j] = 0$ for $i \neq j$. In fact, with the random variables as defined, we have that $\text{Cov}[X_i, X_j] = 1$ for all i, j.

As you saw in this problem, learning more about the distribution of random variables makes it easier to provide tighter bounds on their sums, and correlations across those variables makes it harder. This is a good intuition to have for later in the quarter, where we'll be discussing how different assumptions on hash functions lead to different analyses of data structures.

Section Two: Algorithmic Prerequisites

Problem Three: Binary Search Trees (4 Points)

This one is all coding, so you don't need to write anything here. Make sure to submit your final implementation on myth.

Problem Four: Event Planning (4 Points)

You're trying to figure out what Fun and Exciting Things you'd like to do over the weekend. You download a list of all the local events going on in your area. Each event is tagged with its location, which you can imagine is a point in the 2D plane. (We'll pretend that the world is flat, at least in a small neighborhood around your location. Thanks, multivariable calculus.) You also have your own (x, y) location.

Design a algorithm that, given some number k, returns a list of the k events that are closest to you, sorted by increasing order of distance. Your algorithm should run in time $O(n + k \log k)$, where n is the number of nearby events. Then prove your algorithm is correct and meets the required time bounds.

Some specific details and edge cases to watch for:

- You can assume, for simplicity, that no two events are at the same distance from you.
- By "distance", we mean Euclidean distance. We're already assuming the world is flat, so while we're at it seems pretty reasonable to also ignore things like roads and speed limits. ☺

As a hint, think about the algorithms you studied in CS161 and see if any of them would make for good subroutines.

To make things easier for the grader, we recommend doing the following when writing up your solution:

1. Start off by giving a quick, two-sentence, high-level description of your approach. This makes it easier for the grader to contextualize what it is that you're trying to do.
2. Next, go into more detail. Describe how your algorithm works, one step at a time. Please don't write actual code unless it's exceptionally well-commented and serves a purpose that plain English couldn't. (Trust us – from experience, reading code is often much harder than reading prose!)
3. Write a quick correctness proof. Tell us what, specifically, you're going to prove, then go prove it. Our proof expectations are similar to those for CS161 – write in complete sentences, use mathematical notation when appropriate but not as a stand-in for plain English, etc.
4. Write a runtime analysis. Go at whatever level of detail seems most appropriate.

A note on this problem, and other problems going forward: when measuring runtime in the context of algorithms and data structures, it's important to distinguish between ***deterministic*** and ***randomized*** algorithms. There's a lot of research into how to take *randomized* algorithms with a nice *expected* runtime and convert them into *deterministic* algorithms with a nice *worst-case* runtime. Since this problem set is designed as a warm-up, we'll accept either a deterministic algorithm with a worst-case runtime of $O(n + k \log k)$ or a randomized algorithm with an expected runtime of $O\left(n + k \log k\right)$, though in the future we'll tend to be a bit stricter about avoiding randomness.

Solution: Your solution goes here!