Problem 1: (Practice with Asymptotic Notation)

An essential requirement for understanding scaling behavior is comfort with asymptotic (or ‘big-O’) notation. In this problem, you will prove some basic facts about such asymptotics.

Part (a)

Given any two functions $f(\cdot)$ and $g(\cdot)$, show that $f(n) + g(n) = \Theta(\max\{f(n), g(n)\})$.

Solution: Note – Unless mentioned otherwise, we will always consider functions from the positive integers to the non-negative real numbers. To show $f(n) + g(n) = \Theta(\max\{f(n), g(n)\})$, we need to show $f(n) + g(n) = \Omega(\max\{f(n), g(n)\})$ and $f(n) + g(n) = O(\max\{f(n), g(n)\})$.

First, since the functions are non-negative, we have that $f(n) + g(n) \geq f(n)$ and $f(n) + g(n) \geq g(n)$ – combining these, we get that $f(n) + g(n) \geq \max\{f(n), g(n)\}$ for all $n$; thus $f(n) + g(n) = \Omega(\max\{f(n), g(n)\})$. On the other hand, we also have that $f(n) + g(n) \leq 2\max\{f(n), g(n)\}$ for all $n$; thus $f(n) + g(n) = O(\max\{f(n), g(n)\})$. This completes the proof.

Part (b)

An algorithm $ALG$ consists of two tunable sub-algorithms $ALG_A$ and $ALG_B$, which have to be executed serially (i.e., one run of $ALG$ involves first executing $ALG_A$ followed by $ALG_B$). Moreover, given any function $f(n)$, we can tune the two algorithms such that one run of $ALG_A$ takes time $O(f(n))$ and $ALG_B$ takes time $O(n/f(n))$. How should we choose $f$ to minimize the overall runtime of $ALG$ (i.e., to ensure the runtime of $ALG$ is $O(h(n))$ for the smallest-growing function $h$)?

How would your answer change if $ALG_A$ and $ALG_B$ could be executed in parallel, and we have to wait for both to finish?

Solution: Since the two algorithms are run sequentially, the total runtime is $O(f(n) + n/f(n))$ – from the previous part, we have that this is same as $O(\max\{f(n), n/f(n)\})$. Now, in order to minimize this, it is clear we need to set $f(n)$ such that both parts are equal. Thus, we should choose $f(n) = \sqrt{n}$ and thus $h(n) = \sqrt{n}$.

In case the two ran in parallel, the runtime would now be $O(\max\{f(n), n/f(n)\})$ – clearly this would have the same optimal runtime!

Part (c)

We are given a recursive algorithm which, given an input of size $n$, splits it into 2 problems of size $n/2$, solves each recursively, and then combines the two parts in time $O(n)$. Thus, if $T(n)$ denotes the runtime for the algorithm on an input of size $n$, then we have:

$$T(n) = 2T(n/2) + O(n)$$

Prove that $T(n) = O(n \log n)$.
Hint: Note that for a constant size input, the algorithm takes $O(1)$ time. How many recursions does it require to reduce a problem of size $n$ to constant size subproblems? What is the total runtime overhead at each recursive level?

Solution: We will solve this via an explicit counting argument, which I find instructive in understanding how runtimes accumulate in a recursion. Let $k = \{1, 2, \ldots, K\}$ denote the levels of the recursion tree – here $k = 1$ is the original problem of size $n$, $k = 2$ is the first level of recursion with two subproblems of size $n/2$, and extending this, at level $k$, we have $2^k$ subproblems, each of size $n/2^k$, and at level $K$ the subproblems are of size 1. Now observe the following:

- The subproblems are of size 1 after $K \leq \lceil \log_2 n \rceil$ recursive levels. Moreover, the time taken to solve a subproblem of size 1 is $O(1)$.
- The overhead from a subproblem of size $n$ is $O(n)$ – thus the total overhead at level $k$ is $2^k \cdot O(n/2^k) = O(n)$

Putting this together, we get that $T(n) = O(n \log n)$.

Problem 2: (Some important asymptotes)

Part (a)

In class, we defined the harmonic number $H_n = \sum_{i=1}^{n} 1/i$. Argue that:

$$\int_1^{n+1} \frac{1}{x} \, dx \leq H_n \leq 1 + \int_1^{n} \frac{1}{x} \, dx$$

Thus, prove that $H_n = \Theta(\ln n)$.

Hint: Bound the $1/x$ function from above and below by a step function.

Solution: The idea is to represent the harmonic number as the area of a curve (see Figure 1).

Essentially, we have that $H_n$ is the area under a set of rectangles of length 1 and height $1/i$, $i \in \{1, 2, \ldots, n\}$. Now suppose we have a step function such that $f_n(x) = 1/i$, $x \in [i, i + 1)$ (i.e., the rectangles above the $1/x$ curve). Then we have:

$$H_n \geq \int_1^{n+1} \frac{1}{x} \, dx = \ln(n + 1)$$

On the other hand, if we define $f_t(x) = 1/i$, $x \in [i - 1, i]$, and consider $x \in [2, n]$, we have that:

$$H_n = 1 + \sum_{i=2}^{n} \frac{1}{i} \leq 1 + \int_1^{n} \frac{1}{x} \, dx = 1 + \ln(n)$$

Thus we have that $\ln(n + 1) \leq H_n \leq 1 + \ln n$ and hence $H_n = \Theta(\log n)$. 

2
Part (b)

Next, we try to find the asymptotic growth $n!$. As in the previous part, argue that:

$$\int_1^n \ln x \, dx \leq \ln n! \leq \int_1^{n+1} \ln x \, dx$$

Thus, prove that $n! = \Theta(n \ln n)$.

**Solution:** This proceeds in a very similar fashion as above, except that now $\log x$ is an increasing function. We first compare the function $f_i(x) = \log i, x \in [i, i+1)$ for $x \in [1, n+1]$ to get:

$$\log n! \leq \int_1^{n+1} \ln x \, dx = x \log x - x|_1^{n+1} = (n + 1) \log(n + 1) - n$$

To lower bound, we use the function $f_u(x) = \log i, x \in (i - 1, i]$ for $x \in [1, n]$ to get:

$$\log n! = \sum_{i=2}^n \log i \geq \int_1^n \ln x \, dx = x \log x - x|_1^n = n \log n - n + 1$$

Combining and using the fact that $n = O(n \log n)$, we get that $\log n! = \Theta(n \log n)$.

Part (c)

(Stirling’s approximation) We now improve the estimate in the previous part to get the familiar form of Stirling’s approximation. First, argue that for any integer $i \geq 1$, we have:

$$\int_i^{i+1} \log x \, dx \geq \frac{\log i + \log(i + 1)}{2}$$
Using this, show that:

\[ n! \leq e\sqrt{n} \left( \frac{n}{e} \right)^n \]

**Hint:** Given any \( i > 1 \), where does the line joining the two points \((i, \ln i)\) and \((i + 1, \ln(i + 1))\) lie with respect to the function \( \log x \)?

**Solution:** We again want to use the integral bounding trick – here however, we use the added property that \( \log x \) is a concave function, and hence for any positive integer \( i \), the line segment joining \((i, \log i)\) and \((i + 1, \log(i + 1))\) lies below the curve \( \log x \) for \( x \in [i, i + 1] \). Moreover, the area of the trapezoid bounded by the line segment joining \((i, \log i)\) and \((i + 1, \log(i + 1))\), and the lines \( x = 0, y = i \) and \( y = i + 1 \) is given by \( \frac{\log(i + 1) + \log(i + 1)}{2} \) (recall – the area of a trapezoid is \( \frac{1}{2} \cdot \text{height} \cdot \text{sum of lengths of parallel sides} \)). Thus we have that:

\[ \int_i^{i+1} \log x \, dx \geq \frac{\log i + \log(i + 1)}{2} \]

Now summing up over \( i \in \{1, 2, \ldots, n - 1\} \), we have:

\[ \sum_{i=1}^{n} \int_i^{i+1} \log x \, dx \geq \log n! - \frac{n}{2} \]

However the left hand side is just \( \int_1^n \log x \, dx = n \log n - n + 1 \). Thus we get:

\[ \log n! \leq n \log n - n + 1 + \frac{\log n}{2} \]

Exponentiating both sides, we get:

\[ n! \leq e\sqrt{n} \left( \frac{n}{e} \right)^n \]

**Problem 3: (The Geometric Distribution)**

A random variable \( X \) is said to have a Geometric\( (p) \) distribution if for any integer \( k \geq 1 \), we have \( \Pr[X = k] = p(1 - p)^{k-1} \).

**Part (a)**

Suppose we repeatedly toss a coin which gives HEADS with probability \( p \). Argue that the number of tosses until we see the first HEADS is distributed as Geometric\( (p) \).

**Solution:** Our sample space \( \Omega \) consists of all sequences over the alphabet \{H, T\} that end with H (HEADS) and contain no other H’s, i.e. \( \Omega = \{H, TH, TTH, \ldots\} \). The number of failures \( k - 1 \) before the first success (HEADS) with a probability of success \( p \) is given by: \( \Pr[X = k] = p(1 - p)^{k-1} \) with \( k \) being the total number of tosses including the first HEADS that terminates the experiment. Therefore, the number of tosses until we see the first HEADS is distributed as Geometric\( (p) \).
Part (b)

(Memoryless property) Using the definition of conditional probability, prove that for any integers \(i, k \geq 1\), the random variable \(X\) obeys:

\[
P[X = k + i | X > k] = P[X = i]
\]

Also convince yourself that this follows immediately from the characterization of the Geometric r.v. in Part (a).

Solution: By the definition of conditional probability,

\[
P[X = k + i | X > k] = \frac{P[X = k + i \cap X > k]}{P[X > k]} = \frac{P[X = k + i]}{P[X > k]} = \frac{p(1-p)^{k+i-1}}{(1-p)^k} = p(1-p)^{i-1} = P[X = i].
\]

Note that here we used that \(P[X > k] = (1-p)^k\). The event "\(X > k\)" means that at least \(k + 1\) tosses are required. This is exactly equivalent to saying that the first \(k\) tosses are all TAILS and the probability of this event is precisely \((1-p)^k\).

Part (c)

Show that: (i) \(E[X] = \frac{1}{p}\), and (ii) \(Var[X] = \frac{1-p}{p^2}\)

Hint: Note that by the memoryless property, a Geometric\((p)\) random variable \(X\) is 1 with probability \(p\), and \(1+Y\) with probability \((1-p)\), where \(Y\) also has a Geometric\((p)\) distribution. Now try writing the expectation and variance recursively.

Solution: Note that by the memoryless property, a Geometric\((p)\) random variable \(X\) is 1 with probability \(p\), and \(1+Y\) with probability \((1-p)\), where \(Y\) also has a Geometric\((p)\) distribution. Therefore,

\[
\]

Solving for \(E[X]\), we get \(E[X] = \frac{1}{p}\).

Next, recall that \(Var[X] = E[X^2] - (E[X])^2 = E[X^2] - \frac{1}{p^2}\). So, first we need to calculate \(E[X^2]\).

\[
E[X^2] = E[p \cdot 1 + (1-p) \cdot (1+Y)^2] = p + (1-p)(1+2 \cdot E[Y] + E[Y^2]) = p + (1-p)(1 + 2 \cdot E[X] + E[X^2]) = p + (1-p)(1 + 2 \cdot \frac{1}{p} + E[X^2]).
\]

Simplifying, we get \(E[X^2] = \frac{2-p}{p^2}\), and hence \(Var(X) = E[X^2] - \frac{1}{p^2} = \frac{1-p}{p^2}\).
Problem 4: (Upper Bounds on Collision Probabilities)

Let $X_{m,n}$ denote the number of collisions when $m$ balls are dropped u.a.r. into $n$ bins. In class, we showed that then the expected number of collisions is $\binom{m}{2}/n$. We now upper bound the probability that no collision occurs.

Assume that $n > m$ (clearly this is required for no collisions!). First, using the law of total probability, argue that:

$$P[\text{No collisions when } m \text{ balls dropped u.a.r. in } n \text{ bins}] = \prod_{i=1}^{m-1} \left(1 - \frac{i}{n}\right)$$

Next, using the inequality $e^{-x} \geq (1 - x)$, simplify the above to show:

$$P[\text{No collisions when } m \text{ balls dropped u.a.r. in } n \text{ bins}] \leq e^{-E[X_{m,n}]}$$

**Solution:** Let $D_i$ be the event that there is no collision after having thrown in the $i$-th ball. If there is no collision after throwing in $i$ balls then they must all be occupying different slots, so the probability of no collision upon throwing in the $(i+1)$-st ball is exactly $(n-i)/n$. That is,

$$P[D_{i+1}|D_i] = \frac{n-i}{n}.$$ 

Also note that $P[D_1] = 1$. The probability of no collision at the end of the game can now be computed via

$$P[D_m] = P[D_m|D_{m-1}] \cdot P[D_{m-1}] = \cdots = \prod_{i=1}^{m-1} P[D_{i+1}|D_i] = \prod_{i=1}^{m-1} \left(1 - \frac{i}{n}\right).$$

Note that $i/n \leq 1$. So we can use the inequality $1x \leq e^x$ for each term of the above expression. This means that:

$$P[D_m] \leq \prod_{i=1}^{m-1} \left(e^{-\frac{i}{n}}\right) = e^{-\frac{m(m-1)}{2n}} = e^{-E[X_{m,n}]}.$$ 

Problem 5: (Posterior Confidence in Verifying Matrix Multiplication)

In class, we saw Freivald’s algorithm for checking matrix multiplication, which, given matrices $A, B$ and $C$, returned the following:

- If $AB = C$, then the algorithm always returned TRUE
- If $AB \neq C$, then the algorithm returned TRUE with probability at most $1/2$

Part (a)

Given any $\epsilon > 0$, how many times do we need to run Freivald’s algorithm to be sure that $\{AB = C\}$ with probability greater than $1 - \epsilon$?
Part (b)

Suppose we started with the belief that the events \( \{ AB = C \} \) and \( \{ AB \neq C \} \) were equally likely (i.e., \( P[AB = C] = P[AB \neq C] = 1/2 \)). Moreover, suppose \( k \) independent runs of Freivald’s algorithm all returned TRUE. Then what is our new (or posterior) belief that \( \{ AB = C \} \)?

Solution:

Part (a)

We know that \( P[AB = C] \geq 1 - \frac{1}{2^n} \), therefore, we want \( 1 - \frac{1}{2^n} \geq 1 - \epsilon \). Which means, \( k \geq -\log_2 \epsilon \), and hence \( n = \lceil -\log_2 \epsilon \rceil \).

Part (b)

Let \( I \) be our information that \( k \) independent runs of Freivald’s algorithm all returned TRUE. Now we simply need to use Bayes’ Theorem to find the posterior:

\[
P[AB = C | I] = \frac{P[I | AB = C] P[AB = C]}{P[I | AB = C] P[AB = C] + P[I | AB \neq C] P[AB \neq C]} = \frac{1 \cdot \frac{1}{2} \cdot \frac{1}{2}}{1 + 2^{-k}} = \frac{1}{1 + 2^{-k}}.
\]