CIS 120/160 Review Session
January 22, 2019
6-8PM
HW1 Updates

- Please proofread your LaTeX!!! We don’t want to see ¡ and ¿ when you mean < and >
- Please use the “align” and “align*” environments to center equations, don’t just hit “Enter” for multiple lines of math
- Please use math mode (i.e. $...$)
- Make sure your solutions are coherent
- Keep it brief: try to be more concise while hitting the main points
- Submit written assignments to BOTH Canvas and Gradescope by the deadline (in this case, Monday 1/28 at 11:59PM)
Random Variables

**Random Variables and Expectation** The following problems and concepts will be useful for

- Analyzing the expected runtime of randomized quick-sort
- Analysing the performance of skip-lists
- Computing the expected encoding length for Huffman codes
- Analyzing the performance of HashMaps
Recall that if $X$ is a random variable, the expected value of $X$, denoted $E(X)$ is given by

$$E(X) = \sum_{x} xP(X = x)$$

That is, it is the weighted sum of the possible values $X$ can take on, where the weights are the probabilities (in fact, this is exactly the center-of-mass of the probability distribution).
Also recall that a geometric variable $X$ with parameter $p$ describes the number of trials one has to perform to see a success, where success on each trial occurs with probability $p$ and the trials are independent. Remember, we say that events $A$ and $B$ are independent if $P(A \cap B) = P(A)P(B)$ and similarly, two R.V.’s $X$ and $Y$ are independent if $P(X = i, Y = j) = P(X = i)P(Y = j)$ for all $i, j$. 
Examples of geometric random variables include:

- The number of coin flips needed to see a head is geometric with parameter $p = \frac{1}{2}$
- The number of die rolls needed to see a 3 is geometric with parameter $p = \frac{1}{6}$
Example

**Problem:** Find the expected value of a geometric random variable with parameter $p$. 
If $X \sim Geom(p)$, then

$$P(X = k) = (1 - p)^{k-1} p$$

since the probability the first success occurs on attempt $k$ is the probability that the first $k - 1$ attempts were failures and the $k$-th attempt was a success. Since the trials are independent, we can just take the products of these probabilities. $X$ can take on any positive integer as its values, hence using the definition of expected value:
Solution Cont.

\[ E(X) = \sum_{k=1}^{\infty} kP(X = k) = \sum_{k=1}^{\infty} k(1 - p)^{k-1}p \]

\[ = p \sum_{k=1}^{\infty} k(1 - p)^{k-1} \]

\[ = p \cdot \left( \frac{1}{(1 - (1 - p))^2} \right) \]

\[ = \frac{p}{p^2} \]

\[ = \frac{1}{p} \]
This is rather intuitive: it says that we should expect to see our first head on the second coin flip, and that we should expect to see our first 3 on the sixth roll of a fair die.

(To obtain the identity \( \sum_{k=1}^{\infty} kr^{k-1} = \frac{1}{(1-r)^2} \), simply differentiate both sides of the expression for the sum of an infinite geometric series \( \sum_{k=0}^{\infty} r^k = \frac{1}{1-r} \).)
Problem: Consider a sequence of $n$ tosses of a fair coin. Define a “run” to be a sequence of consecutive tosses that all produce the same result (i.e. heads or tails). For example, if $n = 7$, then the sequence of outcomes

$$H H H T T H T$$

has four runs, namely $(H H H H)$, $(T T)$, $(H)$, and $(T)$. Given a sequence of $n$ tosses of a fair coin, what is the expected number of runs you will see?
Solution

Let $X$ be the random variable denoting the number of runs we see. We wish to compute $E(X)$. Define the indicator random variables

$$X_i = \begin{cases} 
1 & \text{there is a run beginning at index } i \\
0 & \text{otherwise}
\end{cases}$$

Note that $X_1 = 1$ since the first outcome always begins a new run. Moreover, we see that

$$X = X_1 + \ldots + X_n = \sum_{i=1}^{n} X_i$$

and so by the linearity of expectation,

$$E(X) = E \left( \sum_{i=1}^{n} X_i \right) = \sum_{i=1}^{n} E(X_i)$$
Recall that for an indicator random variable $Y$, 
$E(Y) = 1 \cdot P(Y = 1) + 0 \cdot P(Y = 0) = P(Y = 1)$, hence for each $i$, we have 

$$E(X_i) = P(X_i = 1)$$

Now a run begins at index $i$ whenever the $i-1$-th outcome and the $i$-th outcome are different. For $i > 1$, this occurs with probability $\frac{1}{2}$. Hence 

$$E(X_i) = \begin{cases} 
1 & i = 1 \\
\frac{1}{2} & i > 1
\end{cases}$$

Thus 

$$E(X) = \sum_{i=1}^{n} E(X_i) = 1 + \sum_{i=2}^{n} E(X_i) = 1 + \sum_{i=2}^{n} \frac{1}{2} = 1 + \frac{n-1}{2} = \frac{n+1}{2}$$
Example

**Problem:** Find the expected length of an arbitrary run.
Let $X$ be the length of an arbitrary run. The run can either be a run of heads or a run of tails. We thus condition on this case: let $Y$ be the random variable that is 1 if the run is a run of heads and 0 if it is a run of tails. Then $P(Y = 1) = P(Y = 0) = \frac{1}{2}$.

Computing the expectation of $X$ by conditioning on $Y$ gives:

$$E(X) = E(X|Y = 1)P(Y = 1) + E(X|Y = 0)P(Y = 0)$$
Given that $Y = 1$, i.e. that the run is a run of heads, $X$ is a geometric random variable with parameter $p = \frac{1}{2}$, since finding the length of the run starting with a head is equivalent to finding the number of coin tosses we make until seeing the first tail. Hence $E(X|Y = 1) = 2$, since the expected value of a geometric random variable with parameter $p$ is $\frac{1}{p}$. Symmetrically, given that $Y = 0$, $X$ is also a geometric random variable with parameter $p = \frac{1}{2}$. Hence $E(X|Y = 0) = 2$ as well. This gives:

$$E(X) = 2 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2} = 2$$
Solution Cont.

Note that because of the symmetry, we could have assumed WLOG that the run started with heads and proceeded from there. However, in the case of a biased coin, we don’t have symmetry and thus conditioning is the way to go.
Probabilistic Method
The probabilistic method is a technique that uses probability and expectation to prove the existence of certain things. The technique is best illustrated by an example:
Problem: A tournament graph $G$ is a directed graph such that for all pairs of vertices $u, v$, exactly one of $(u, v)$ or $(v, u)$ is an edge in $G$. You can think of the graph as specifying who beat whom when every player in a tournament plays every other player exactly once. Alternatively, you can think of a tournament graph as starting with an undirected complete graph and then assigning directions to each edge.

Fix some $n > 0$. Prove that there exists a tournament graph $G$ on $n$ vertices such that $G$ contains at least $n! \cdot 2^{-(n-1)}$ Hamiltonian paths.
Let $G$ be a randomly chosen tournament graph (that is, choose a tournament graph uniformly at random from all possible tournament graphs on $n$ vertices). Let $X$ denote the number of Hamiltonian paths in $G$. Note that a Hamiltonian path is just a permutation $\sigma$ of the vertices in $G$ such that there is an edge between adjacent vertices, i.e. $(\sigma(i), \sigma(i+1))$ is an edge for all $i = 1 \ldots n - 1$. There are $n!$ possible permutations of vertices. For any given permutation $\sigma$ and any fixed $i \in \{1 \ldots n - 1\}$, the probability that $(\sigma(i), \sigma(i+1))$ is an edge is $G$ is 1/2. Hence the probability that $\sigma$ specifies a Hamiltonian path in $G$ is $\frac{1}{2^{n-1}}$. Hence, the expected number of Hamiltonian paths in $G$ is

$$E(X) = n! \cdot \frac{1}{2^{n-1}}$$
Since $G$ was chosen at random, and this is the expected number of Hamiltonian paths, it follows that there must exist *some* tournament graph with at least $n! \cdot \frac{1}{2^{n-1}}$ Hamiltonian paths. This completes the proof.
Note how the solution is completely non-constructive: it relies solely on expectation and probability to prove that there exists a graph with $n! \cdot 2^{-(n-1)}$ Hamiltonian paths.
Definitions

A **graph** $G = (V, E)$ is a set of **vertices** (also called **nodes**) $V$ together with a set of **edges** $E \subseteq V \times V$, where $V \times V$ denotes the cartesian product of $V$ with itself. We denote an edge from $u$ to $v$ as $(u, v)$. A graph can be undirected, in which case we consider the edges $(u, v)$ and $(v, u)$ to be the same edge, or it can be directed, in which case we distinguish $(u, v)$ from $(v, u)$. In CIS 121, we don’t worry about so-called “self-loops”, i.e. edges of the form $(v, v)$. The **degree** of a vertex $v$, denoted $\text{deg}(v)$ is the number of edges incident on it (for directed graphs, we distinguish between in-degree and out-degree).
A **path** from $u$ to $v$ is a sequence $u = v_0, v_1, ..., v_n = v$ such that $(v_i, v_{i+1}) \in E$ for all $0 \leq i < n$. The **length** of a path is the number of edges in the path. A **simple path** is a path containing distinct vertices, i.e. for all $i \neq j$, $v_i \neq v_j$.

A **cycle** is a sequence $u = v_0, v_1, ..., v_n = u$, essentially a path with the same start and end point. Graphs that contain cycles are called **cyclic** while graphs that don’t contain cycles are called **acyclic**.
A **connected** graph is an undirected graph such that for any two vertices $u$ and $v$, there is a path from $u$ to $v$. A **strongly connected** graph extends this definition to directed graphs: for any two vertices $u$ and $v$, there is a path from $u$ to $v$ and a path from $v$ to $u$. A **(strongly) connected component** of a graph $G$ is a subgraph of $G$ that is (strongly) connected and maximal, i.e. it is (strongly) connected and not contained in any larger (strongly) connected component.
**Definitions Cont.**

**Special Types of Graphs**
A **tree** is an undirected, connected acyclic graph. Some special defining properties of trees are given below:

- A tree with \( n \) nodes is connected and has exactly \( n - 1 \) edges.
- For any vertices \( u \) and \( v \) in a tree \( T \), there is a unique path from \( u \) to \( v \).
- Tree’s are minimially connected: that is, a tree \( T \) is connected, but the removal of any edge from \( T \) will disconnect it.
- A tree is acyclic, but adding *any* edge to the tree will create a cycle.

Any of the above can be used as the definition of a tree. A vertex with degree 1 is called a **leaf**. A **forest** is an acyclic, undirected graph. A forest is just the union of one or more trees.
A **binary tree** is a tree such that every vertex has degree at most 2. Usually we consider the tree to be **rooted** at some root vertex $r$. The **height** of a binary tree is the number of edges in the longest path from the root node to a leaf. Also, instead of considering the neighbors of vertices in a binary tree, we normally call them the left/right children. The **ancestors**, **descendants**, and **parents** of nodes in a binary tree are defined exactly how you think they should be.
Example

**Problem:** Prove that any tree $T$ with $n$ vertices, $n \geq 2$ has at least two nodes with degree 1.
Consider a maximal path in $T$. Call this path $P : v_1, ..., v_n$. I claim $v_1$ and $v_n$ both have degree 1. Suppose for the sake of contradiction that this weren’t the case, and WLOG let $\deg(v_1) > 1$. Then $v_1$ has a neighbor in $T$ other than $v_2$, call this neighbor $u$. If $u$ is on the path $P$, then $T$ contains a cycle, contradicting the fact that $T$ is a tree. If $u$ is not on the path $P$, then we could define $P' : u, v_1, ..., v_n$ to be path containing $P$ but that is strictly larger than $P$. But this contradicts the maximality of $P$. Hence in all cases we reach a contradiction. The case for when $\deg(v_n) > 1$ is symmetric. Hence $v_1$ and $v_n$ have degree 1, completing the proof.
Induction
Induction is a proof technique that relies on the following logic: If I have a statement $P(n)$ such that the statement is true for $n = 1$, i.e. $P(n) = TRUE$, AND I know that whenever $P(k)$ is true for any $k \geq 1$, then $P(k + 1)$ is also true, then the statement must be true for all integers $n \geq 1$. The typical intuitive examples of induction are:

- Suppose I have a line of dominoes. If I know I can knock down the first domino, and I know that if the $k$-th domino falls, it will knock over the $k + 1$-th domino, then all the dominoes will fall.

- Suppose I have a ladder. If I know that I can get on the ladder, and I know that for each step of the ladder it is possible to get to the next step, then I can climb the whole ladder.
Induction

There are plenty of other examples. In general, an inductive proof of a claim $P$ consists of proving that a **base case (BC)** holds (usually $P(0)$ or $P(1)$), making an **induction hypothesis (IH)** that assumes that $P(k)$ is true for some value $k$ that is greater than or equal to the base case, and lastly an **induction step (IS)** which uses the IH to prove that $P(k + 1)$ holds. Please note that induction can only be used to prove properties about integers—you can’t use induction to prove statements about real numbers.
**Problem:** Prove via induction that \( \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \).
Solution

We proceed via induction on $n$.

**Base Case:** The base case occurs when $n = 1$. In this case,

\[
\sum_{i=1}^{1} i = 1 \quad \text{and} \quad \frac{1(1+1)}{2} = 1,
\]

and so the claim holds in the base case.

**Induction Hypothesis:** Assume that for some integer $k \geq 1$ we have

\[
\sum_{i=1}^{k} i = \frac{k(k+1)}{2}.
\]
Induction Step: We must show that \( \sum_{i=1}^{k+1} i = \frac{(k+1)(k+2)}{2} \). Indeed:

\[
\sum_{i=1}^{k+1} i = (k + 1) + \sum_{i=1}^{k} i \\
= (k + 1) + \frac{k(k + 1)}{2} \quad \text{by the induction hypothesis} \\
= \frac{2(k + 1)}{2} + \frac{k(k + 1)}{2} \\
= \frac{2(k + 1) + k(k + 1)}{2} \\
= \frac{(k + 1)(k + 2)}{2}
\]
Solution Cont.

Note how we used the induction hypothesis in the induction step. If you find yourself doing a proof by induction without invoking the induction hypothesis, you are probably doing it incorrectly (or induction is not necessary).
Strong Induction

The above describes what is called “weak induction”. Another variant, called **strong induction** assumes in the induction hypothesis that not only does $P(k)$ hold for some $k$ that is greater than or equal to the base case (say, $b$), but that $P(i)$ also holds for all $b \leq i \leq k$. 
Example

**Problem:** Show that the number of vertices in a full binary tree of height $h$ is at most $2^{h+1} - 1$. 
Solution

We proceed via induction on $h$, the height of the binary tree.

**Base Case:** A tree of height $h = 0$ consists of a single node, hence has size 1. Since $2^{0+1} - 1 = 2 - 1 = 1$, the claim holds in the base case.

**(Strong) Induction Hypothesis:** Assume that for a binary tree of height $h$ for some $h \geq 0$ that the number of vertices in the tree is at most $2^{h+1} - 1$. 
Induction Step: Consider a tree $T$ of height $h + 1$ with root $x$. We must show that $T$ has at most $2^{(h+1)+1} - 1 = 2^{h+2} - 1$ nodes. Consider the left and right subtrees of $x$, call them $l$ and $r$ respectively. Since $T$ has height $h + 1$, the left and right subtrees have height at most $h$ (hence why we need to use strong induction). By the IH, the left and right subtrees have at most $2^{h+1} - 1$ nodes each. Hence the total number of nodes in the binary tree is at most $(2^{h+1} - 1) + (2^{h+1} - 1) + 1 = 2 \cdot (2^{h+1} - 1) + 1 = 2^{h+2} - 2 + 1 = 2^{h+1} - 1$ nodes, which is exactly what we wanted to show.
Example

**Problem:** Show that any tree with $n$ vertices has $n - 1$ edges.
Solution

We proceed via induction on $n$, the number of vertices.

**Base Case:** A tree with $n = 1$ vertices is just a single node with no edges, and hence the claim is satisfied.

**Induction Hypothesis:** Assume that for some $n, n \geq 1$ that every tree with $n$ vertices has $n - 1$ edges.
Induction Step: Consider a tree $T$ with $n + 1$ vertices. We must show $T$ has $n$ edges. By the result of problem 2 (see above), we know that $T$ has at least two leaves. Pick one of these leaves, call it $l$, and remove it from the graph to form a new graph $T'$. Note that since there was only one edge incident on $l$, the graph $T'$ has $n$ vertices. Moreover, $T'$ is connected since we only removed a leaf and is acyclic since removing a node cannot create a cycle. Hence $T'$ is a tree. By the induction hypothesis, $T'$ has $n - 1$ edges. Since we formed $T'$ from $T$ by removing a single vertex $l$ and a single edge, this implies $T$ has $n$ edges, which is exactly what we wanted to show.
120 Concepts and Common Programming Issues
Common Issues

▶ You can’t instantiate an interface
  ▶ Interfaces describe methods
  ▶ Classes implement those methods
▶ Recursive functions need base cases
  ▶ If you get a stack overflow error, you probably forgot a base case
  ▶ Recursion is powerful but requires a “leap of faith” in the recursive step
▶ Objects can be null, primitives can’t
  ▶ Watch out for NPE’s
  ▶ Make sure you are testing equality correctly
  ▶ Careful with pointers!
▶ Pointers, primitives, and new
“Static”

Static variables are shared between all instances of an object.

- If one instance updates a static field, all instances feel the effect.

Static methods are similar—this means they cannot reference/modify non-static fields or call non-static methods.
Static vs. Dynamic

Static:
- Can be called without an instance of the class (e.g. Math.random(), Collections.sort())
- Cannot reference non-static fields or call non-static methods

Dynamic:
- Must have an instance of the class
- Can call any static or non-static methods
- Can modify the internal state of the calling object
Parameterization

Can parameterize classes to make them more generic

Rather than have separate classes for an IntList, a StringList, a DoubleList, etc., can create a generic class List<E>

E stands for an arbitrary type

- Good because it’s generic
- Bad because you can’t assume anything about it (except that it is an Object)
- A specific type will be provided upon instantiation (e.g. List<Integer> l = new LinkedList<Integer>();)
- In this case, everywhere you see an “E”, replace with “Integer” to understand the behavior
The End