Please print your name:

This three-hour exam is due by Thursday December 16th at 9pm.

This exam is closed book, closed notes, closed videos, closed internet. Once you have started the exam, you may not consult any of the course materials or external resources.

Pick any 5 of the following 6 questions to solve in either directly on this exam (if you print it out) or on separate sheets of paper. Please upload a single PDF of your solutions to the corresponding Canvas assignment. You must circle which questions you wish to be graded.

You must show all your work to receive full credit. Simply stating the answer is not enough.

Please sign the honor code: I have neither given nor received unauthorized aid on this assignment.

---

**Sets**

<table>
<thead>
<tr>
<th>symbol</th>
<th>meaning</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \cap B$</td>
<td>set intersection</td>
<td>set containing all elements which are elements of both $A$ and $B$</td>
</tr>
<tr>
<td>$A \cup B$</td>
<td>set union</td>
<td>set containing all elements which are elements of $A$ or $B$ or both</td>
</tr>
<tr>
<td>$\overline{A}$</td>
<td>complement</td>
<td>set of everything which is not an element of $A$</td>
</tr>
<tr>
<td>$A \setminus B$</td>
<td>set-minus</td>
<td>set containing all elements of $A$ which are not elements of $B$</td>
</tr>
<tr>
<td>$</td>
<td>A</td>
<td>$</td>
</tr>
</tbody>
</table>

**Logarithms**

- $\log(x^a) = a \log(x)$ where $a$ is a constant.
- $\log_b(c) = x$ means $b^x = c$. Also, $a = b^\log_b(a)$.
- $\log_b(a \cdot c) = \log_b(a) + \log_b(c)$. Also, $\log_b(a/c) = \log_b(a) - \log_b(c)$.
- A logarithm in one base $a$ can be converted to another base $b$ using $\log_b(x) = \frac{\log_a(x)}{\log_a(b)}$.

**Summations**

- Adding the same value $n$ times: $\sum_{i=1}^{n} c = c \cdot n$ where $c$ is a constant.

- Arithmetic series: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

- Geometric series: $\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}$, for $|r| \neq 1$.

**Asymptotics**

Given functions $f, g$ which map $x$ from $\mathbb{R} \to \mathbb{R}$, then $f$ is $O(g)$ if $\lim_{x \to \infty} \left| \frac{f(x)}{g(x)} \right| < \infty$. 

Graph theorems

Theorem 1. The sum of the vertex degrees equals twice the number of edges in a graph: \( \sum_{v \in V} \deg(v) = 2|E| \).

Theorem 2. A graph with maximum degree \( k \) can be colored with \( (k+1) \) colors.

Theorem 3. A tree with \( n \) vertices has \( n - 1 \) edges.

Theorem 4. A full \( k \)-ary tree is a \( k \)-ary tree in which every internal vertex (which includes the root, unless it is the only vertex) has \( k \) children. Here are a few properties of full \( k \)-ary trees:

1. The number of vertices \( n \) in a full \( k \)-ary tree with \( i \) internal vertices is \( n = ki + 1 \).
2. The number of leaves \( \ell \) in a full \( k \)-ary tree with \( i \) internal vertices is \( \ell = i(k - 1) + 1 \).

Combinations & Permutations

The number of ways to choose \( k \) items out of \( n \) is

\[
C(n, k) = \binom{n}{k} = \frac{n!}{k!(n-k)!} = C(n, n-k).
\]

The number of ways to permute \( k \) items out of \( n \) is

\[
P(n, k) = \frac{n!}{(n-k)!} = C(n, k) \cdot k!.
\]

The number of ways to pick the location of \( n \) stars, or \( k-1 \) bars, from \( n+k-1 \) positions is

\[
\binom{n+k-1}{n} = \binom{n+k-1}{k-1}.
\]

Probability

- Given a sample space \( S \), the probability of an event \( E \) occurring is: \( p(E) = \frac{|E|}{|S|} \).
- Given events \( A \) and \( B \), the probability of \( A \) given \( B \) is
  \[
p(A \mid B) = \frac{p(A \cap B)}{p(B)}, \quad p(B) \neq 0.
  \]
- The events \( A \) and \( B \) are independent if-and-only if: \( p(A \cap B) = p(A) \cdot p(B) \).
- The expectation (expected value) of a random variable \( X : S \to \mathbb{R} \) is: \( E[X] = \sum_{s \in S} X(s)p(s) \).
- For an indicator random variable \( X : S \to \{0,1\} \) and \( A \subseteq S \), \( E[X] = \sum_{s \in A} p(s) = p(A) \),
  where \( X(s) = 1 \) for \( s \in A \), \( X(s) = 0 \) for \( s \notin A \).
- For random variables \( X_1 \) and \( X_2 \), \( E[X_1 + X_2] = E[X_1] + E[X_2] \).
- For a random variable \( X \), \( E[aX + b] = aE[X] + b \) for \( a, b \in \mathbb{R} \).

Tree method

For divide-and-conquer recurrences of the form:

\[
f(n) = a \cdot f\left(\frac{n}{b}\right) + c \cdot n^d
\]

where \( n = b^k \) for some \( k \in \mathbb{Z}^+, a \geq 1, b > 1, b \in \mathbb{Z}, c > 0, d \geq 0 \) and \( a, c, d \in \mathbb{R} \):

\[
f(n) = \begin{cases} 
O(n^d) & \text{if } a < b^d \\
O(n^d \log n) & \text{if } a = b^d \\
O(n \log_a^d) & \text{if } a > b^d.
\end{cases}
\]
Problem 1 [10 points]

For each of the following, you can leave your answer in terms of binomial coefficients, or factorials. In order to receive full credit, please explain every step in your reasoning.

(a) [3 points] How many paths are there from the point (0,0) to the point (4,6) if there is an impassable boulder at (0,2)? Assume that you can only go in the +x or +y direction in a single step.

(b) [4 points] Suppose you have 5 fair dice. What is the probability that two dice show the same value, and that the remaining three dice show different values?

(c) [3 points] How many bit-strings of length 10 have at least seven 1s? Recall that a bit-string of length $n$ is a string of $n$ bits, where each bit is either 0 or 1.
Problem 2 [10 points]

Consider the following algorithm for drawing a tree (perhaps with Python using the turtle). All the drawing occurs in the forward function, and we want to analyze the number of operations performed in this algorithm. Suppose that the call to forward is linear in \( \ell \), that is \( F(\ell) = c \cdot \ell \) for some constant \( c \). You may ignore the calls to turn_left, turn_right and backward in your analysis. Define the problem size as the length of the branch that is drawn during each call to draw_tree. A sample output for draw_tree is shown on the right.

\[
\text{draw_tree}(\ell)
\]

\[
\begin{align*}
\text{input:} & \quad \ell \text{ (length of branch to draw)} \\
1 & \quad \alpha = 20 \text{ (angle of each branch)} \\
2 & \quad \text{if } \ell < 1 \quad \# \text{ base case} \\
3 & \quad \text{return} \\
4 & \quad \text{else} \quad \# \text{ recursive case} \\
5 & \quad 6 \quad \text{forward}(\ell) \quad \# \text{ draw moving forward} \\
7 & \quad 8 \quad \text{turn_left}(\alpha) \\
9 & \quad 10 \quad \text{draw_tree}(\ell/2) \\
10 & \quad 11 \quad \text{turn_left}(\alpha) \\
11 & \quad 12 \quad \text{draw_tree}(\ell/2) \\
12 & \quad 13 \quad \text{turn_right}(3\alpha) \\
13 & \quad 14 \quad \text{draw_tree}(\ell/2) \\
14 & \quad 15 \quad \text{turn_right}(\alpha) \\
15 & \quad 16 \quad \text{draw_tree}(\ell/2) \\
16 & \quad 17 \quad \text{turn_left}(2\alpha) \\
17 & \quad 18 \quad \text{backward}(\ell) \quad \# \text{ return to start}
\end{align*}
\]

(a) How many subproblems (new branches) are created in the recursive call of draw_tree?

(b) What is the size of each subproblem relative to the length of the current branch, \( \ell \)?

(c) Develop a recurrence relation for the number of operations performed in draw_tree.

(d) Use the Tree Method to determine a big-oh bound on the number of operations performed.

(e) If the number of operations performed by forward is instead modeled as \( F(\ell) = c \cdot \ell^2 \) (again, for some constant \( c \)), how would your answer to part (d) change?
Problem 3 [10 points]

Suppose you determined that the following function is the slowest part of a much larger codebase. You have been given the task of rewriting the result of this function into a closed-form mathematical expression to speed it up.

(a) Express the result of this function (the output of Line 4) as a recurrence relation.

(b) What are the base cases (for $n = 0$ and $n = 1$)?

(c) Determine a closed-form expression for $\text{function}(n)$ in terms of $n$ only (i.e. your answer should no longer depend on recursive calls to $\text{function}$).

(d) Prove, using a proof by induction, that your function indeed returns the correct value. You may either work with the algorithm in question, or your recurrence relation from part (a).

\begin{verbatim}
function(n)
  input: n ∈ N
  1  if n < 2    # base cases
  2      return 3(1 − n)
  3  else      # recursive case
  4      return 5 × function(n − 1) − 4 × function(n − 2)
\end{verbatim}
Problem 4 [10 points]

Let’s play a game! Pick a number between 1 and 6. Now roll three fair, independent dice. Calculate your earnings for this round using the following rules:

- If your number does not come up at all, you lose a dollar.
- If your number comes up once, you win a dollar.
- If your number comes up twice, you win two dollars.
- If your number comes up three times, you win four dollars.

We want to analyze your average earnings in each round of this game.

(a) Let $X_i$ be an indicator random variable that is equal to 1 if you have $i$ matches with your chosen number after rolling the three dice. Express the earnings per round as a weighted sum of these indicator random variables.

(b) What is the probability of having $i$ matches with your chosen number? It might help to consider the size of the sample space, as well as the number of outcomes in which you achieve (i) 0 matches? (ii) 1 match? (iii) 2 matches? or (iv) 3 matches with your chosen number.

(c) Apply linearity of expectation to calculate the average earnings during each round of this game. Based on your analysis, should you play this game?
Problem 5 [10 points]

Let’s determine whether Piper will go clam-hunting, given that she is hungry. We will assume the following:

- There is a 99/100 chance Piper will go hunting for clams.
- If Piper goes hunting for clams, there is a 9/10 chance she is hungry.
- If Piper doesn’t go hunting for clams, there is a 1/20 chance she is hungry.

What is the probability that Piper goes hunting for clams, given that she is hungry?

*Hint: build the probability tree.*
Problem 6 [10 points]

Prove that a tree with 2 or more vertices is 2-colorable (the chromatic number is 2).