Weekly Assignment 2
Total: 100

CS 2500: Algorithms

Due Date: September 10, 2024 at 11.59 PM

Instructions

- Submit your solutions by the deadline specified above.
- Ensure that your work is your own.
- Write your answers clearly and show all your work.
- If you have any questions, ask during recitations or office hours.

Problems

1. **Sums of Geometric Progressions.** Use mathematical induction to prove this formula for the sum of a finite number of terms of a geometric progression with initial term a and common ratio r:

$$\sum_{j=0}^{n} ar^j = a + ar + ar^2 + \cdots + ar^n = \frac{ar^{n+1} - a}{r - 1} \quad \text{when } r \neq 1$$

   where n is a nonnegative integer. **[20 points]**

2. **An Inequality for Harmonic Numbers.** The harmonic numbers H_j are defined by:

$$H_j = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{j}$$

   Prove by mathematical induction that:

$$H_{2^n} \geq 1 + \frac{n}{2}$$

   for all nonnegative integers n. **[20 points]**

3. Give a recursive algorithm for finding a mode of a list of integers. (A mode is an element in the list that occurs at least as often as every other element.) **[10 points]**

Generating Valid Binary Strings Using Recursion

This is a problem that tests your understanding of recursion, string manipulation, and algorithmic efficiency. Companies like Meta and Amazon often ask this question to evaluate your problem-solving skills and ability to work with recursion and strings.

- You must submit both the source code and a brief report that explains your approach, the algorithms used, and any challenges encountered.
- Ensure your code is well-documented, with comments explaining the purpose of key sections and functions.
- Your code will be evaluated on both correctness and efficiency, as well as clarity and organization.

Part 1: Coding Exercise [20 points]

Task

Write a function `generate_valid_strings(n)` that takes a positive integer n and returns all valid binary strings of length n, where a binary string is considered valid if every substring of length 2 contains at least one "1".

Constraints

The function should return all valid strings of length n in any order.

Examples

- **Example 1:**
  - **Input:** `n = 3`
  - **Output:** `["010","011","101","110","111"]`
  - **Explanation:**
    The valid strings of length 3 are: `"010"`, `"011"`, `"101"`, `"110"`, and `"111"`. All substrings of length 2 in these strings contain at least one "1".
- **Example 2:**
  - **Input:** `n = 1`
  - **Output:** `["0","1"]`
  - **Explanation:** The valid strings of length 1 are simply `"0"` and `"1"`.
- **Example 3:**
  - **Input:** `n = 2`
  - **Output:** `["01","10","11"]`
  - **Explanation:** The valid strings of length 2 are `"01"`, `"10"`, and `"11"`. Each of these contains at least one "1" in every substring of length 2.

Part 2: Analysis [30 points]

1. Analyze the time complexity of your implementation. [**5 points**]
2. What is the space complexity of your solution? Discuss whether there are trade-offs between time and space complexity in your approach. [**5 points**]

3. How does your solution ensure that each valid binary string is generated exactly once? What specific techniques or steps do you use to avoid duplicates? [**5 points**]

4. Describe how your algorithm handles the following edge cases:

   (a) $n = 1$. [**2 points**]

   (b) $n = 2$. [**2 points**]

   (c) Very large values of n, such as $n = 20$. [**4 points**]

5. How does your algorithm perform as the value of n increases towards higher limits (e.g., $n = 20$ or $n = 30$)? What are the potential bottlenecks, and how might you address them? [**4 points**]

6. Suppose the problem was extended to require that every substring of length 3 contains at least one "'1". How would your approach change to handle this new constraint? [**3 points**]