Weekly Assignment 4
Total: 100

CS 2500: Algorithms

Due Date: October 29, 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.

Weighted Median

Consider n elements $x_1, x_2, \ldots, x_n$ with positive weights $w_1, w_2, \ldots, w_n$ such that

$$\sum_{i=1}^{n} w_i = 1.$$

The **weighted (lower) median** is an element x_k satisfying

$$\sum_{x_i < x_k} w_i < \frac{1}{2} \quad \text{and} \quad \sum_{x_i > x_k} w_i \leq \frac{1}{2}.$$

Example

Consider the following elements x_i and weights w_i:

i	1	2	3	4	5	6	7
x_i	3	8	2	5	4	1	6
w_i	0.12	0.35	0.025	0.08	0.15	0.075	0.2

For these elements, the median is $x_5 = 4$, but the weighted median is $x_7 = 6$.

To see why the weighted median is x_7, observe that the elements less than x_7 are x_1, x_3, x_4, x_5, and x_6, and the sum $w_1 + w_3 + w_4 + w_5 + w_6 = 0.45$, which is less than $\frac{1}{2}$. Furthermore, only element x_2 is greater than x_7, and $w_2 = 0.35$, which is no greater than $\frac{1}{2}$.

Problems

1. Argue that the median of $x_1, x_2, \ldots, x_n$ is the weighted median of the x_i with weights $w_i = \frac{1}{n}$ for $i = 1, 2, \ldots, n$. [**10 points**]
2. Show how to compute the weighted median of n elements in $O(n \log n)$ worst-case time using sorting. [**10 points**]

Finding the i Largest Elements in a List

Task. You are given a set of n numbers, and you wish to find the i largest in sorted order using a comparison-based algorithm.

1. **Method 1: Sort the Numbers and List the i Largest**
   (a) Describe the steps involved in this method. [**2 points**]
   (b) What is the worst-case time complexity of this approach? Justify your answer. [**3 points**]
2. **Method 2: Use an Order-Statistic Algorithm**
   (a) Explain how you can use an order-statistic algorithm to find the i largest elements. [**4 points**]
   (b) What are the steps involved in this method after identifying the ith largest element? [**4 points**]
   (c) Analyze the overall worst-case time complexity of this method. Can this approach offer better performance than Method 1? Explain why or why not, considering different values of i. [**8 points**]
3. **Comparison and Analysis**
   (a) Compare the two methods in terms of time complexity. Under what conditions is Method 2 more efficient than Method 1? [**5 points**]
   (b) Suggest a scenario where Method 1 would be preferable over Method 2, and explain your reasoning. [**4 points**]

Searching a Value in a Sorted Matrix

This problem tests your ability to develop efficient algorithms that can navigate structured data efficiently. It is a popular question in technical interviews at leading tech companies like Google, Microsoft, and Amazon, assessing your problem-solving skills, particularly in software engineering and data science roles.

- 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.

Coding Exercise [20 points]

Task

Write a function `searchMatrix(matrix, target)` that takes a 2D integer matrix `matrix` and an integer `target`, and returns `true` if `target` is present in the matrix, or `false` otherwise. The function must run in $O(m + n)$ time, where m is the number of rows and n is the number of columns.

Constraints

- `m == matrix.length`
- `n == matrix[i].length`
- $1 \leq m$, $mleq 300$
- $-10^9 leq \text{matrix}[i][j] \leq 10^9$
- All the integers in each row are sorted in ascending order.
- All the integers in each column are sorted in ascending order.
- $-10^9 leq \text{target} \leq 10^9$

Examples

- **Example 1:**
  - **Input:** `matrix = [[1, 4, 7, 11, 15], [2, 5, 8, 12, 19], [3, 6, 9, 16, 22], [10, 13, 14, 17, 24], [18, 21, 23, 26, 30]]`, `target = 5`
  - **Output:** `true`
  - **Explanation:** The target value 5 is found in the matrix.
- **Example 2:**
  - **Input:** `matrix = [[1, 4, 7, 11, 15], [2, 5, 8, 12, 19], [3, 6, 9, 16, 22], [10, 13, 14, 17, 24], [18, 21, 23, 26, 30]]`, `target = 20`
  - **Output:** `false`
  - **Explanation:** The target value 20 is not present in the matrix.

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. Describe the key steps and techniques that allow your algorithm to efficiently search for the target value. [**5 points**]
4. Explain how your algorithm handles the following edge cases:
   (a) The target is smaller than the smallest element in the matrix. [**2 points**]
   (b) The target is larger than the largest element in the matrix. [**2 points**]
   (c) The matrix contains only one row or one column. [**4 points**]
5. How does your algorithm perform when m and n are at their maximum limits (e.g., $m = 300$ and $n = 300$)? Identify any potential performance bottlenecks, and suggest ways to address them. [**4 points**]
6. Suppose the problem was extended to search for multiple target values in the same matrix. How would your approach change to handle this new constraint? [**3 points**]