<https://leetcode.com/problems/maximum-area-of-a-piece-of-cake-after-horizontal-and-vertical-cuts/description/>

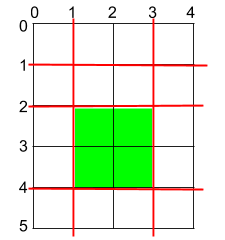
You are given a rectangular cake of size h x w and two arrays of integers horizontalCuts and verticalCuts where:

horizontalCuts[i] is the distance from the top of the rectangular cake to the ith horizontal cut and similarly, and

verticalCuts[j] is the distance from the left of the rectangular cake to the jth vertical cut.

Return *the maximum area of a piece of cake after you cut at each horizontal and vertical position provided in the arrays* horizontalCuts *and* verticalCuts. Since the answer can be a large number, return this **modulo** 109 + 7.

**Example 1:**

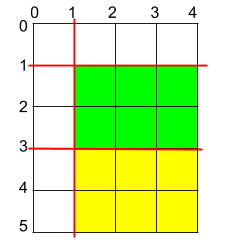


**Input:** h = 5, w = 4, horizontalCuts = [1,2,4], verticalCuts = [1,3]

**Output:** 4

**Explanation:** The figure above represents the given rectangular cake. Red lines are the horizontal and vertical cuts. After you cut the cake, the green piece of cake has the maximum area.

**Example 2:**



**Input:** h = 5, w = 4, horizontalCuts = [3,1], verticalCuts = [1]

**Output:** 6

**Explanation:** The figure above represents the given rectangular cake. Red lines are the horizontal and vertical cuts. After you cut the cake, the green and yellow pieces of cake have the maximum area.

**Example 3:**

**Input:** h = 5, w = 4, horizontalCuts = [3], verticalCuts = [3]

**Output:** 9

**Constraints:**

2 <= h, w <= 109

1 <= horizontalCuts.length <= min(h - 1, 105)

1 <= verticalCuts.length <= min(w - 1, 105)

1 <= horizontalCuts[i] < h

1 <= verticalCuts[i] < w

All the elements in horizontalCuts are distinct.

All the elements in verticalCuts are distinct.

**Attempt 1: 2025-06-29**

**Solution 1: Math (10 min)**

class Solution {

    public int maxArea(int h, int w, int[] horizontalCuts, int[] verticalCuts) {

        int MOD = (int)(1e9 + 7);

        // Sort the cuts to easily find maximum gaps

        Arrays.sort(horizontalCuts);

        Arrays.sort(verticalCuts);

        int maxHorizontalGap = Math.max(verticalCuts[0], w - verticalCuts[verticalCuts.length - 1]);

        int maxVerticalGap = Math.max(horizontalCuts[0], h - horizontalCuts[horizontalCuts.length - 1]);

        // Calculate maximum horizontal segment

        for(int i = 1; i < verticalCuts.length; i++) {

            maxHorizontalGap = Math.max(maxHorizontalGap, verticalCuts[i] - verticalCuts[i - 1]);

        }

        // Calculate maximum vertical segment

        for(int i = 1; i < horizontalCuts.length; i++) {

            maxVerticalGap = Math.max(maxVerticalGap, horizontalCuts[i] - horizontalCuts[i - 1]);

        }

        // Return the product modulo 10^9 + 7

        return (int)((long) maxHorizontalGap \* maxVerticalGap % MOD);

    }

}

Time Complexity: O(n)

Space Complexity: O(1)

**Refer to Deepseek**

This problem requires finding the maximum area of a rectangular piece of cake after making horizontal and vertical cuts. Here's an efficient Java solution with explanation:

**Solution Code**

class Solution {

public int maxArea(int h, int w, int[] horizontalCuts, int[] verticalCuts) {

// Sort the cuts to easily find maximum gaps

Arrays.sort(horizontalCuts);

Arrays.sort(verticalCuts);

// Calculate maximum horizontal segment

int maxH = Math.max(horizontalCuts[0], h - horizontalCuts[horizontalCuts.length - 1]);

for (int i = 1; i < horizontalCuts.length; i++) {

maxH = Math.max(maxH, horizontalCuts[i] - horizontalCuts[i - 1]);

}

// Calculate maximum vertical segment

int maxW = Math.max(verticalCuts[0], w - verticalCuts[verticalCuts.length - 1]);

for (int i = 1; i < verticalCuts.length; i++) {

maxW = Math.max(maxW, verticalCuts[i] - verticalCuts[i - 1]);

}

// Return the product modulo 10^9 + 7

return (int)((long)maxH \* maxW % 1000000007);

}

}

**Explanation**

**Approach**

**Sort the Cuts**:

First sort both horizontal and vertical cuts to easily find the maximum gaps between cuts.

**Calculate Maximum Gaps**:

For horizontal cuts:

The maximum gap is either:

Between the start (0) and first cut

Between consecutive cuts

Between last cut and end (h)

Similarly for vertical cuts (between 0 and w)

**Compute Maximum Area**:

The maximum area is the product of the maximum horizontal gap and maximum vertical gap

Return the result modulo 10^9 + 7 as required by the problem

**Key Points**

**Sorting**: Essential to efficiently find maximum gaps between cuts

**Edge Cases**: Need to consider gaps before first cut and after last cut

**Modulo Operation**: Handles potential integer overflow for large values

**Time Complexity**

O(n log n + m log m) where n and m are the number of horizontal and vertical cuts respectively

Dominated by the sorting steps

**Space Complexity**

O(1) - Only uses constant extra space beyond input

**Example Walkthrough**

For input:

h = 5, w = 4

horizontalCuts = [1,2,4]

verticalCuts = [1,3]

After sorting (already sorted in this case)

Horizontal gaps:

1-0 = 1

2-1 = 1

4-2 = 2

5-4 = 1

maxH = 2

Vertical gaps:

1-0 = 1

3-1 = 2

4-3 = 1

maxW = 2

Maximum area = 2 \* 2 = 4

This solution efficiently finds the maximum possible cake area by focusing on the largest segments created by the cuts.