**Maximum Area**

**I. Problem**

* *Maximum Area of a Piece of Cake After Horizontal and Vertical Cuts*.
* 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**.

**Chart

Description automatically generated**For example:

**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.

**II. Solve problem**

We can find the max height and the max width separately. Our final answer will be **maxHeight** \* **maxWidth**. Each height and width are defined by the distance between 2 cuts.

In the above example, the max height of 2 is defined by the distance between cuts 2 and 4 (4 - 2 = 2). To find all heights and widths, we must first **sort** our inputs **horizontalCuts** and **verticalCuts**. This will ensure that all of the cuts that are beside each other on the cake are also beside each other in the array.

Then, we can iterate through the sorted inputs one at a time and find each height or width by simply taking the difference between two adjacent cuts.

One thing to be careful about is the edges. For cuts in the middle, the distance is defined by the difference between two cuts. However, for the edges, they are defined by the cake's dimensions.

* The top-most cut's height will be equal to **horizontalCuts[0]**, while the bottom-most cut's height will be equal to

**h - horizontalCuts[horizontalCuts.length - 1]**.

* The left-most cut's width will be equal to **verticalCuts[0]**, while the right-most cut's width will be equal to

**w - verticalCuts[verticalCuts.length - 1]**.

**Algorithm**:

1. Sort both **horizontalCuts** and **verticalCuts** in ascending order (we will employ quick sort).
2. Initialize a variable **maxHeight** as the larger of the top and bottom edge: **maxHeight = max(horizontalCuts[0]**,

**h - horizontalCuts[horizontalCuts.length - 1])**

1. Iterate through **horizontalCuts** starting from index 1 (skip the 0th index since it represents the edge cut, which we accounted for in the previous step). At each iteration, find the height defined by the ith cut and the nearest cut above, **horizontalCuts[i] - horizontalCuts[i - 1]**. Update **maxHeight** if necessary.
2. Initialize a variable **maxWidth** as the larger of the left and right edge: **maxWidth = max(verticalCuts[0]**,

**w - verticalCuts[verticalCuts.length - 1])**

1. Iterate through **verticalCuts** starting from index 1. At each iteration, find the width defined by the ith cut and the nearest cut to the left,

**verticalCuts[i] - verticalCuts[i - 1]**. Update **maxWidth** if necessary.

1. Our maximum area is **maxHeight \* maxWidth**. Don't forget the modulo 109 + 7, be careful of overflow. Return the maximum area.