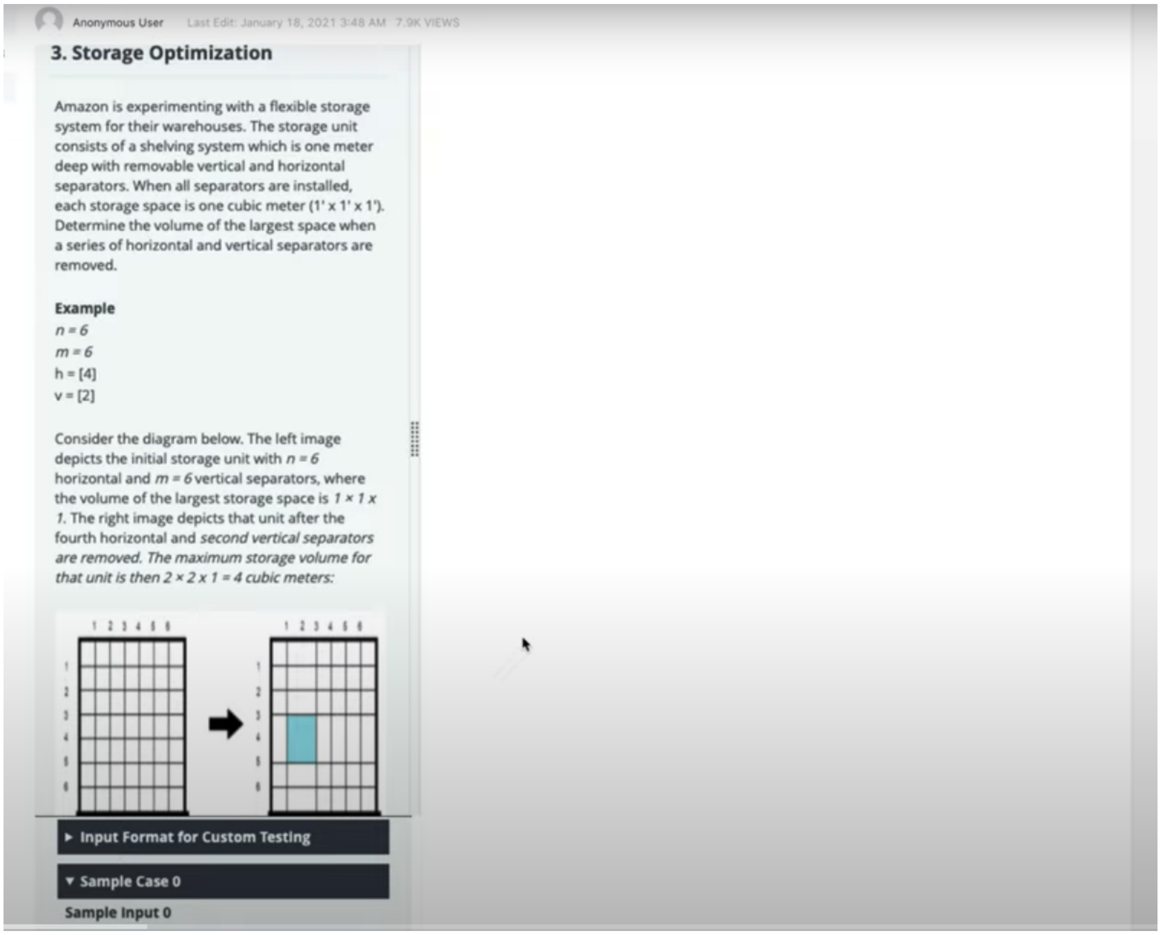
OA

- **Cloudfront caching 地里有**
  union find

- **Storage optimization**



Anonymous User Last Edit: January 18, 2021 3:48 AM 7.9K VIEWS

3. Storage Optimization

Amazon is experimenting with a flexible storage system for their warehouses. The storage unit consists of a shelving system which is one meter deep with removable vertical and horizontal separators. When all separators are installed, each storage space is one cubic meter (1' x 1' x 1'). Determine the volume of the largest space when a series of horizontal and vertical separators are removed.

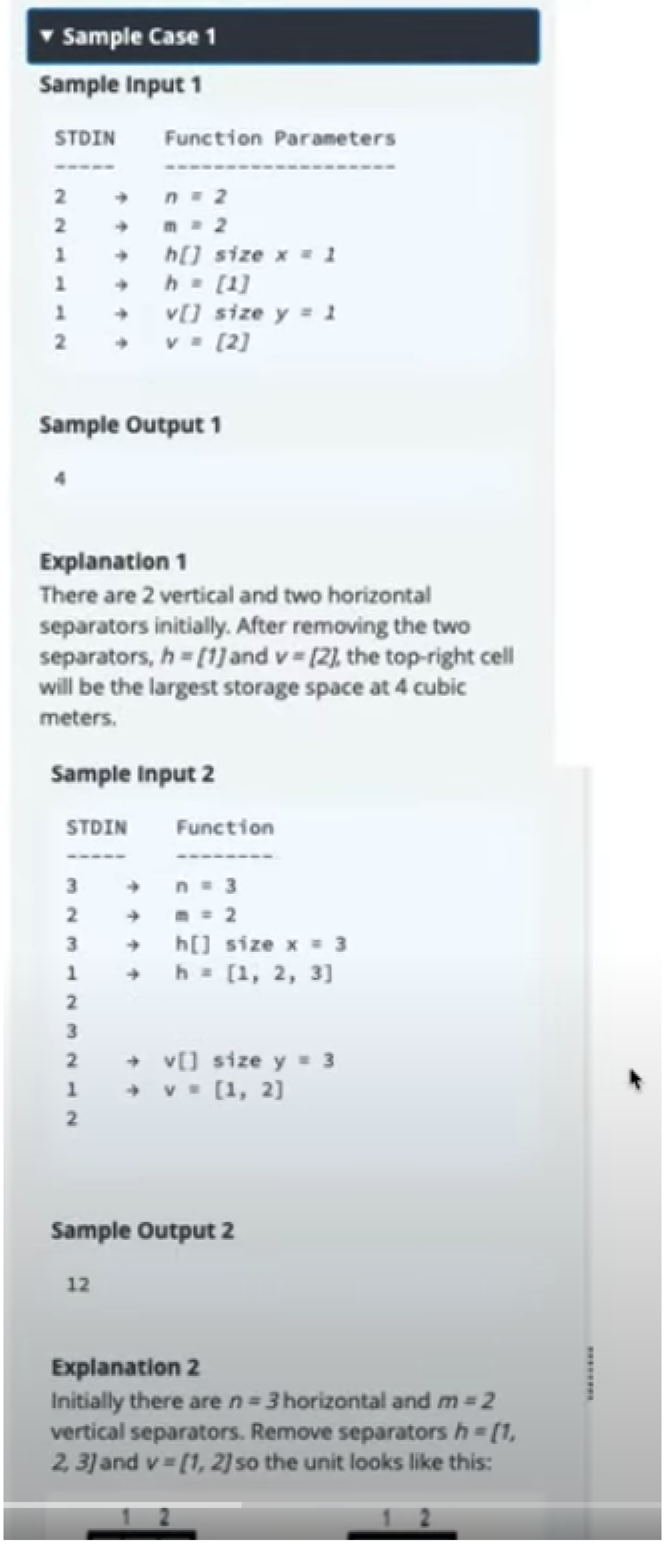
Example
n = 6
m = 6
h = [4]
v = [2]

Consider the diagram below. The left image depicts the initial storage unit with *n = 6* horizontal and *m = 6* vertical separators, where the volume of the largest storage space is *1 × 1 x 1*. The right image depicts that unit after the fourth horizontal and *second vertical separators are removed. The maximum storage volume for that unit is then 2 × 2 x 1 = 4 cubic meters:*

▸ Input Format for Custom Testing

▾ Sample Case 0

Sample Input 0



▼ **Sample Case 1**

Sample Input 1

```
STDIN    Function Parameters
-----    -------------------
2    →   n = 2
2    →   m = 2
1    →   h[] size x = 1
1    →   h = [1]
1    →   v[] size y = 1
2    →   v = [2]
```

Sample Output 1

```
4
```

Explanation 1

There are 2 vertical and two horizontal separators initially. After removing the two separators, *h = [1]* and *v = [2]*, the top-right cell will be the largest storage space at 4 cubic meters.

Sample Input 2

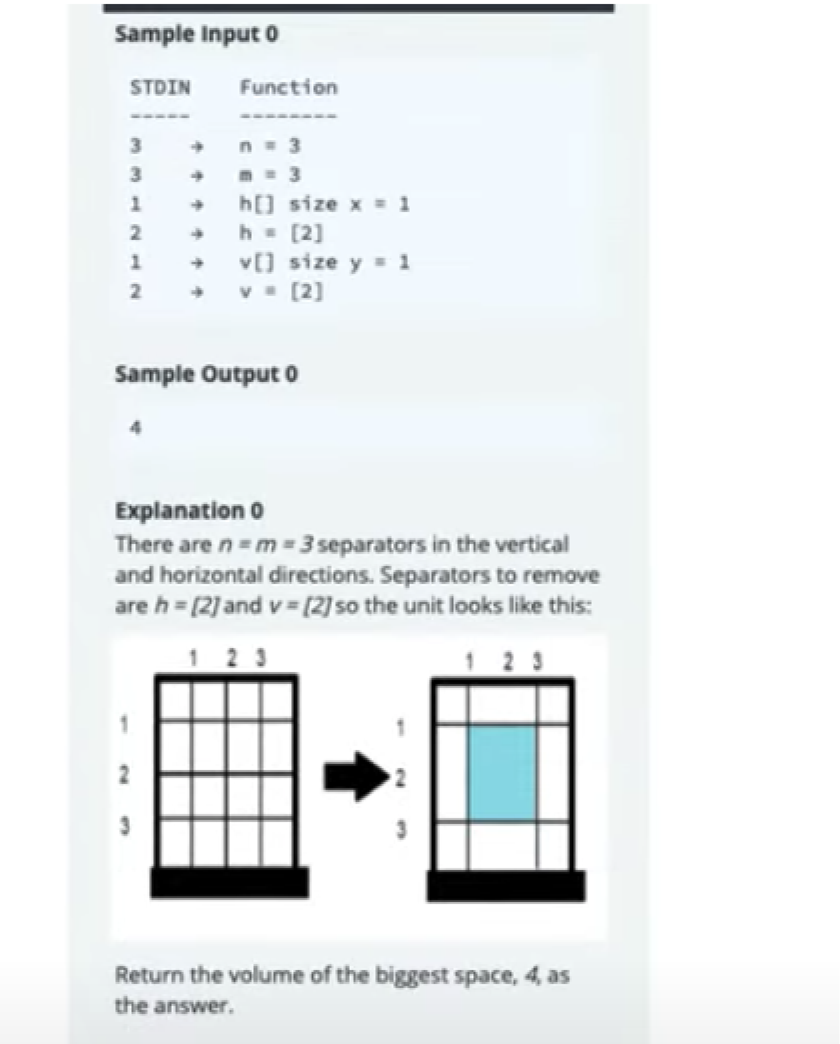
```
STDIN    Function
-----    --------
3    →   n = 3
2    →   m = 2
3    →   h[] size x = 3
1    →   h = [1, 2, 3]
2
3
2    →  v[] size y = 3
1    →  v = [1, 2]
2
```

Sample Output 2

```
12
```

Explanation 2

Initially there are *n = 3* horizontal and *m = 2* vertical separators. Remove separators *h = [1, 2, 3]* and *v = [1, 2]* so the unit looks like this:



Sample Input 0

```
STDIN     Function
-----     --------
3    →    n = 3
3    →    m = 3
1    →    h[] size x = 1
2    →    h = [2]
1    →    v[] size y = 1
2    →    v = [2]
```

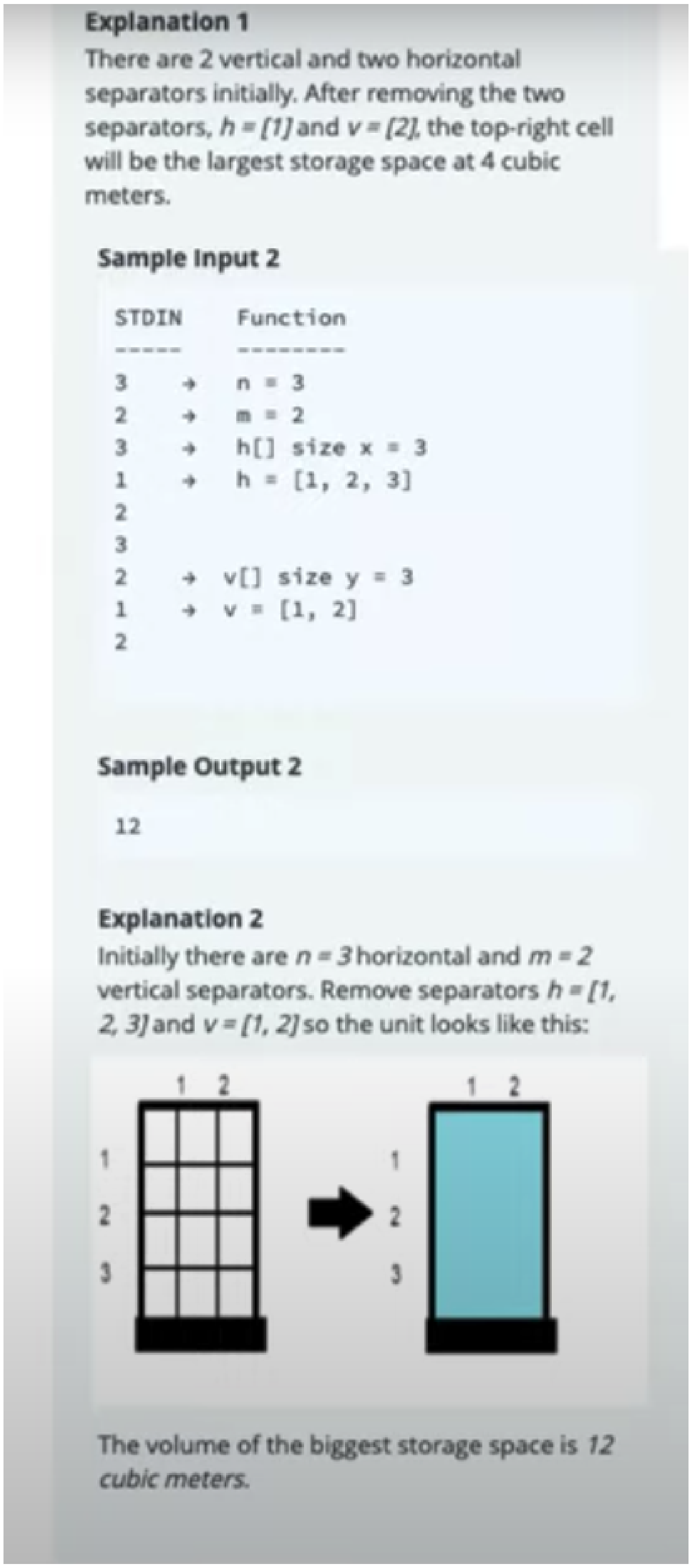
Sample Output 0

```
4
```

Explanation 0

There are *n = m = 3* separators in the vertical and horizontal directions. Separators to remove are *h = [2]* and *v = [2]* so the unit looks like this:

Return the volume of the biggest space, *4*, as the answer.



Explanation 1

There are 2 vertical and two horizontal separators initially. After removing the two separators, *h = [1]* and *v = [2]*, the top-right cell will be the largest storage space at 4 cubic meters.

Sample Input 2

```
STDIN    Function
-----    --------
3    →   n = 3
2    →   m = 2
3    →   h[] size x = 3
1    →   h = [1, 2, 3]
2
3
2    →  v[] size y = 3
1    →  v = [1, 2]
2
```

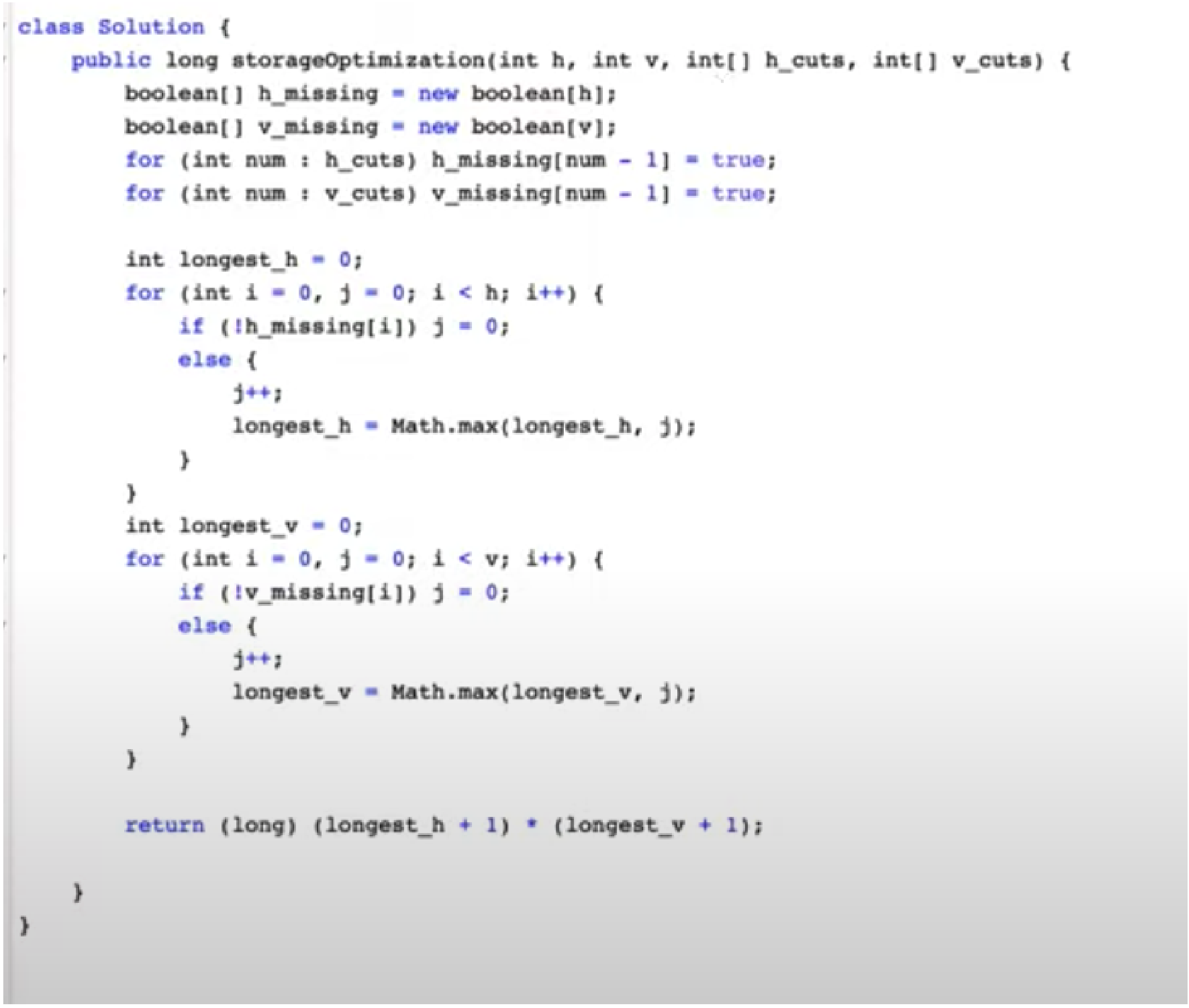
Sample Output 2

```
12
```

Explanation 2

Initially there are *n = 3* horizontal and *m = 2* vertical separators. Remove separators *h = [1, 2, 3]* and *v = [1, 2]* so the unit looks like this:

The volume of the biggest storage space is *12 cubic meters.*



```
class Solution {
    public long storageOptimization(int h, int v, int[] h_cuts, int[] v_cuts) {
        boolean[] h_missing = new boolean[h];
        boolean[] v_missing = new boolean[v];
        for (int num : h_cuts) h_missing[num - 1] = true;
        for (int num : v_cuts) v_missing[num - 1] = true;

        int longest_h = 0;
        for (int i = 0, j = 0; i < h; i++) {
            if (!h_missing[i]) j = 0;
            else {
                j++;
                longest_h = Math.max(longest_h, j);
            }
        }
        int longest_v = 0;
        for (int i = 0, j = 0; i < v; i++) {
            if (!v_missing[i]) j = 0;
            else {
                j++;
                longest_v = Math.max(longest_v, j);
            }
        }

        return (long) (longest_h + 1) * (longest_v + 1);

    }
}
```

- **Shopping options**
  地里有
  Treemap?

  or

  4 个 arrays 分成两组，用一组和二组之和建一个 sum array - [sum1, sum2 …. sumN]
  sum array 排序
  Loop through A3 and A4,
  Binary Search last item that is <= target from sum array

- **Robot Bounded In Circle (LC 1041)**
  https://leetcode.com/problems/robot-bounded-in-circle/

- https://leetcode.com/problems/minimum-cost-to-connect-sticks/

- **Amazon fresh deliveries**
  https://leetcode.com/discuss/interview-question/1033264/amazon-oa-1-year-experienced-for-sde1

  Priority queue

- **Demolition Robot**
  https://leetcode.com/discuss/interview-question/1033264/amazon-oa-1-year-experienced-for-sde1
  bfs

- **Reorder log**
  https://leetcode.com/problems/reorder-data-in-log-files

- **Prime air route**（题库能找 ）
  https://leetcode.com/discuss/interview-question/1025705/Amazon-or-OA-or-Prime-Air-time
  优化到nlogn，通过binarysearch和tuple+map

  1, group by value
  2. Sort by value
  3. Two pointers. For A, start from 0, Fob B, start from end
  4. Loop through A[i] and B[j] to build result

- **Optimizing Box Weight**

```
 def minimalHeaviestSetA(arr):
        arr.sort(reverse=True)
        total =sum(arr)
        res =[]
        sumA = 0
        for i in range(0,len(arr)):
            res.append(arr[i])
            sumA +=arr[i]
            sumB = total - sumA
            if sumA > sumB:
                break
        return res[::-1]
```

- **Shopping pattern**
  https://aonecode.com/amazon-online-assessment-shopping-patterns
  https://www.youtube.com/watch?v=U196c9aQq5c

- **Number of swaps in algorithm**
  https://algo.monster/problems/amazon_oa_number_of_swaps_to_sort
  https://leetcode.com/problems/count-of-smaller-numbers-after-self/
  https://www.geeksforgeeks.org/counting-inversions/

- **Range sum**
  https://leetcode.com/problems/count-of-range-sum/