Brick Wall

There is a brick wall in front of you. The wall is rectangular and has several rows of bricks. The bricks have the same height but different width. You want to draw a vertical line from the **top** to the **bottom** and cross the **least** bricks.

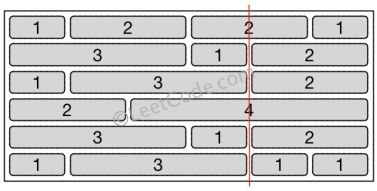
The brick wall is represented by a list of rows. Each row is a list of integers representing the width of each brick in this row from left to right.

If your line go through the edge of a brick, then the brick is not considered as crossed. You need to find out how to draw the line to cross the least bricks and return the number of crossed bricks.

You cannot draw a line just along one of the two vertical edges of the wall, in which case the line will obviously cross no bricks.

Example:

```
Input:
[[1,2,2,1],
 [3,1,2],
 [1,3,2],
 [2,4],
 [3,1,2],
 [1,3,1,1]]
Output: 2
Explanation:
```



Note:

1. The width sum of bricks in different rows are the same and won't exceed INT_MAX.
2. The number of bricks in each row is in range [1,10,000]. The height of wall is in range [1,10,000]. Total number of bricks of the wall won't exceed 20,000.

Solution 1

```
public class Solution {
    public int leastBricks(List < List < Integer >> wall) {
        HashMap < Integer, Integer > map = new HashMap < > ();
        for (List < Integer > row: wall) {
            int sum = 0;
            for (int i = 0; i < row.size() - 1; i++) {
                sum += row.get(i);
                if (map.containsKey(sum))
                    map.put(sum, map.get(sum) + 1);
                else
                    map.put(sum, 1);
            }
        }
        int res = wall.size();
        for (int key: map.keySet())
            res = Math.min(res, wall.size() - map.get(key));
        return res;
    }
}
```

written by vinod23 original link here

Solution 2

For each potential cut position - which is at the edge of any brick, I am counting the number of brick edges for all rows. Note that we need to use hash map to only track potential (not all) cuts. If bricks are very wide, you'll get MLE if you store all cut positions.

```
int leastBricks(vector<vector<int>>& wall) {
    unordered_map<int, int> edges;
    auto min_bricks = wall.size();
    for (auto row : wall)
        for (auto i = 0, width = 0; i < row.size() - 1; ++i) // skip last brick
            min_bricks = min(min_bricks, wall.size() - (++edges[width += row[i]]));
    return min_bricks;
}
```

written by votrubac original link here

Solution 3

```
class Solution(object):
    def leastBricks(self, wall):
        """
        :type wall: List[List[int]]
        :rtype: int
        """
        d = collections.defaultdict(int)
        for line in wall:
            i = 0
            for brick in line[:-1]:
                i += brick
                d[i] += 1
        # print len(wall), d
        return len(wall)-max(d.values()+[0])
```

written by szhu3210 original link here