Python/C++ Solution with explanation

dawyjing ★ 656 Last Edit: September 8, 2019 12:56 AM 1.1K VIEWS

19

If we create the first square matrix, the big matrix will just be the copies of this one. (translation copies)
The value of each location in the square matrix will appear at multiple locations in the big matrix, count them.
Then assign the ones in the square matrix with more occurances with 1.

```
class Solution:
    def maximumNumberOfOnes(self, width: int, height: int, sideLength: int, maxOnes: int) -> int:
        res = []
        for i in range(sideLength):
            for j in range(sideLength):
                res += [((width - i - 1) // sideLength + 1) * ((height - j - 1) // sideLength + 1)]
        res = sorted(res,reverse = True)
        return sum(res[:maxOnes])
```

Or 1-liner just for fun..

```
class Solution:
    def maximumNumberOfOnes(self, width: int, height: int, sideLength: int, maxOnes: int) -> int:
        return sum(sorted([((width-i-1)//sideLength+1)*((height-j-1)//sideLength+1) for i in range(sideLength) for j in range(sideLength)],rever
```

Or in C++

```
class Solution {
public:
    int maximumNumberOfOnes(int width, int height, int sideLength, int maxOnes) {
    std::vector<int> res;
    for(int i = 0;i < sideLength; i++){
        for(int j =0; j < sideLength; j++) res.push_back(((width-i-1)/sideLength+1)*((height-j-1)/sideLength+1));
    }
    sort(res.begin(), res.end(), greater<int>());
    int ans = 0;
    for(int i = 0;i < maxOnes; i++) ans += res[i];
    return ans;
    }
};
```

For large sideLength, we can use heap to store list res, but here sorting the whole list is faster than heapq implementation.

Comments: 3 Best | Most Votes | Newest to Oldest | Oldest to Newest

Type comment here... (Markdown is supported)

Post

ryx ★ 86 September 8, 2019 4:26 AM

how to prove your greedy solution?

4 Reply

yuanb10 ★ 664 September 7, 2019 10:17 PM

Thanks for sharing. Great solution.

I am curious how do you get to the step where you find "the big matrix will just be the copies of this one. ". During the contest, I got stuck figuring out the cases where sub squares overlap with each other thus could not model the problem properly.

3 Show 1 reply Reply

bl2003 ★ 15 September 8, 2019 12:34 AM

The same idea in Java:

```
class Solution {
    public int maximumNumberOfOnes(int width, int height, int sideLength, int maxOnes) {
        int M = sideLength * sideLength;
        if (maxOnes >= M) return width * height;
        int[] res = new int[M];
        for (int i = 0; i < sideLength; i++) {
            for (int j = 0; j < sideLength; j++) {
                res[i * sideLength + j] += ((width - i - 1) / sideLength + 1) * ((height - j - 1) / sideLength + 1);
            }
```

Read More

1 Show 1 reply Reply