Our Solution(s)

Run Code

Your Solutions

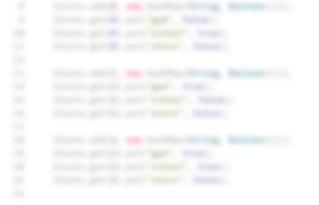
Solution 3

Run Code

```
Solution 1 Solution 2
 1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
   import java.util.*;
   class Program {
     // O(b^2*r) time \mid O(b) space - where b is the number of blocks and
      // requirements
      public static int apartmentHunting(List<Map<String, Boolean>> block
        int[] maxDistancesAtBlocks = new int[blocks.size()];
        Arrays.fill(maxDistancesAtBlocks, Integer.MIN_VALUE);
10
11
        for (int i = 0; i < blocks.size(); i++) {</pre>
12
          for (String req : reqs) {
14
            int closestReqDistance = Integer.MAX_VALUE;
15
            for (int j = 0; j < blocks.size(); j++) {</pre>
16
             if (blocks.get(j).get(req)) {
17
                closestReqDistance = Math.min(closestReqDistance, distance
18
19
20
            maxDistancesAtBlocks[i] = Math.max(maxDistancesAtBlocks[i], cl
21
22
23
        return getIdxAtMinValue(maxDistancesAtBlocks);
24
25
26
      public static int getIdxAtMinValue(int[] array) {
27
        int idxAtMinValue = 0;
28
        int minValue = Integer.MAX_VALUE;
29
        for (int i = 0; i < array.length; i++) {</pre>
30
          int currentValue = array[i];
31
          if (currentValue < minValue) {</pre>
           minValue = currentValue;
33
            idxAtMinValue = i;
```

```
Solution 1 Solution 2
1 import java.util.*;
3 class Program {
    public static int apartmentHunting(List<Map<String, Boolean>> blocks
     // Write your code here.
8 }
```

Our Tests Custom Output Submit Code



Run or submit code when you're ready.