Solution 1 Solution 2

Run Code

Our Solution(s)

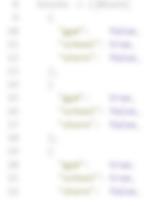
Run Code

```
Your Solutions
```

```
Solution 1  Solution 2  Solution 3

1  package main
2
3  type Block map[string]bool
4
5  func ApartmentHunting(blocks []Block, reqs []string) int {
6    // Write your code here.
7  return -1
```

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
    package main
    import "math"
    type Block map[string]bool
    // O(br) time | O(br) space - where b is the number of blocks
10 // and r is the number of requirements.
11 func ApartmentHunting(blocks []Block, reqs []string) int {
       \label{eq:minDistancesFromBlocks} \begin{array}{l} \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \end{array} \\ \vdots \\ \begin{bmatrix} \vdots \\ \end{bmatrix} \\ \begin{array}{l} \text{int} \\ \{ \} \\ \\ \end{array}
12
13
       for _, req := range reqs {
14
         minDistancesFromBlocks = append(minDistancesFromBlocks,
15
             getMinDistances(blocks, req))
16
17
       maxDistancesAtBlocks := getMaxDistancesAtBlocks(blocks, minDistances
18
19
       var optimalBlockIdx int
20
       smallestMaxDistance := math.MaxInt32
21
       \begin{tabular}{lll} \textbf{for} & \textbf{i,} & \textbf{currentDistance} & \textbf{:= range} & \textbf{maxDistancesAtBlocks} & \textbf{\{} \\ \end{tabular}
22
         if currentDistance < smallestMaxDistance {</pre>
            smallestMaxDistance = currentDistance
24
             optimalBlockIdx = i
26
27
       return optimalBlockIdx
28 }
29
30 func getMinDistances(blocks []Block, req string) []int {
31
       minDistances := make([]int, len(blocks))
32
       closestReq := math.MaxInt32
33
       for i := range blocks {
```



Run or submit code when you're ready.