```go
// 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
// and r is the number of requirements.
func ApartmentHunting(blocks []Block, reqs []string) int {
  minDistancesFromBlocks := [][]int{}
  for _, req := range reqs {
    minDistancesFromBlocks = append(minDistancesFromBlocks,
      getMinDistances(blocks, req))
  }
  maxDistancesAtBlocks := getMaxDistancesAtBlocks(blocks, minDistancesFromBlocks)

  var optimalBlockIdx int
  smallestMaxDistance := math.MaxInt32
  for i, currentDistance := range maxDistancesAtBlocks {
    if currentDistance < smallestMaxDistance {
      smallestMaxDistance = currentDistance
      optimalBlockIdx = i
    }
  }
  return optimalBlockIdx
}

func getMinDistances(blocks []Block, req string) []int {
  minDistances := make([]int, len(blocks))
  closestReq := math.MaxInt32
  for i := range blocks {
    if val, found := blocks[i][req]; found && val {
      closestReq = i
    }
    minDistances[i] = distanceBetween(i, closestReq)
  }

  for i := len(blocks) - 1; i >= 0; i-- {
    if val, found := blocks[i][req]; found && val {
      closestReq = i
    }
    minDistances[i] = min(minDistances[i], distanceBetween(i, closestReq))
  }
  return minDistances
}

func getMaxDistancesAtBlocks(blocks []Block, minDistancesFromBlocks [][]int) []int {
  maxDistancesAtBlocks := make([]int, len(blocks))
  for i := range blocks {
    minDistancesAtBlock := []int{}
    for _, distances := range minDistancesFromBlocks {
      minDistancesAtBlock = append(minDistancesAtBlock, distances[i])
    }
    maxDistancesAtBlocks[i] = max(minDistancesAtBlock)
  }
  return maxDistancesAtBlocks
}

func distanceBetween(a, b int) int {
  if a > b {
    return a - b
  }
  return b - a
}

func min(a, b int) int {
  if a < b {
    return a
  }
  return b
}

func max(array []int) int {
  if len(array) == 0 {
    return 0
  }

  max := array[0]
  for i := 1; i < len(array); i++ {
    if array[i] > max {
      max = array[i]
    }
  }
  return max
}
```