

Solution 1Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3
4 #include <vector>
5 #include <unordered_map>
6 #include <climits>
7 #include <algorithm>
8 #include <cmath>
9
10 using namespace std;
11
12 vector<int> getMinDistances(vector<unordered_map<string, bool>> blocks,
13                             string req);
14 vector<int> getMaxDistancesAtBlocks(vector<unordered_map<string, bool>> blocks,
15                                     vector<vector<int>> minDistancesFromBlocks);
16 int getIdxAtMinValue(vector<int> array);
17 int distanceBetween(int a, int b);
18
19 // O(br) time | O(br) space - where b is the number of blocks and r is the
20 // number of requirements
21 int apartmentHunting(vector<unordered_map<string, bool>> blocks,
22                       vector<string> reqs) {
23     vector<vector<int>> minDistancesFromBlocks;
24     for (string req : reqs) {
25         minDistancesFromBlocks.push_back(getMinDistances(blocks, req));
26     }
27     vector<int> maxDistancesAtBlocks =
28         getMaxDistancesAtBlocks(blocks, minDistancesFromBlocks);
29     return getIdxAtMinValue(maxDistancesAtBlocks);
30 }
31
32 vector<int> getMinDistances(vector<unordered_map<string, bool>> blocks,
33                             string req) {
34     vector<int> minDistances(blocks.size());
35     int closestReqIdx = INT_MAX;
36     for (int i = 0; i < blocks.size(); i++) {
37         if (blocks[i][req])
38             closestReqIdx = i;
39         minDistances[i] = distanceBetween(i, closestReqIdx);
40     }
41     for (int i = blocks.size() - 1; i >= 0; i--) {
42         if (blocks[i][req])
43             closestReqIdx = i;
44         minDistances[i] = min(minDistances[i], distanceBetween(i, closestReqIdx));
45     }
46     return minDistances;
47 }
48
49 vector<int>
50 getMaxDistancesAtBlocks(vector<unordered_map<string, bool>> blocks,
51                         vector<vector<int>> minDistancesFromBlocks) {
52     vector<int> maxDistancesAtBlocks(blocks.size());
53     for (int i = 0; i < blocks.size(); i++) {
54         vector<int> minDistancesAtBlock;
55         for (vector<int> distances : minDistancesFromBlocks) {
56             minDistancesAtBlock.push_back(distances[i]);
57         }
58         maxDistancesAtBlocks[i] =
59             *max_element(minDistancesAtBlock.begin(), minDistancesAtBlock.end());
60     }
61     return maxDistancesAtBlocks;
62 }
63
64 int getIdxAtMinValue(vector<int> array) {
65     int idxAtMinValue = 0;
66     int minValue = INT_MAX;
67     for (int i = 0; i < array.size(); i++) {
68         int currentValue = array[i];
69         if (currentValue < minValue) {
70             minValue = currentValue;
71             idxAtMinValue = i;
72         }
73     }
74     return idxAtMinValue;
75 }
76
77 int distanceBetween(int a, int b) { return abs(a - b); }
```

