```
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

// O(b^2*r) time | O(b) space - where b is the number of blocks and r is the number of requirements
function apartmentHunting(blocks, reqs) {
  const maxDistancesAtBlocks = new Array(blocks.length).fill(-Infinity);
  for (let i = 0; i < blocks.length; i++) {
    for (const req of reqs) {
      let closestReqDistance = Infinity;
      for (let j = 0; j < blocks.length; j++) {
        if (blocks[j][req]) {
          closestReqDistance = Math.min(closestReqDistance, distanceBetween(i, j));
        }
      }
      maxDistancesAtBlocks[i] = Math.max(maxDistancesAtBlocks[i], closestReqDistance);
    }
  }
  return getIdxAtMinValue(maxDistancesAtBlocks);
}

function getIdxAtMinValue(array) {
  let idxAtMinValue = 0;
  let minValue = Infinity;
  for (let i = 0; i < array.length; i++) {
    const currentValue = array[i];
    if (currentValue < minValue) {
      minValue = currentValue;
      idxAtMinValue = i;
    }
  }
  return idxAtMinValue;
}

function distanceBetween(a, b) {
  return Math.abs(a - b);
}

exports.apartmentHunting = apartmentHunting;

```