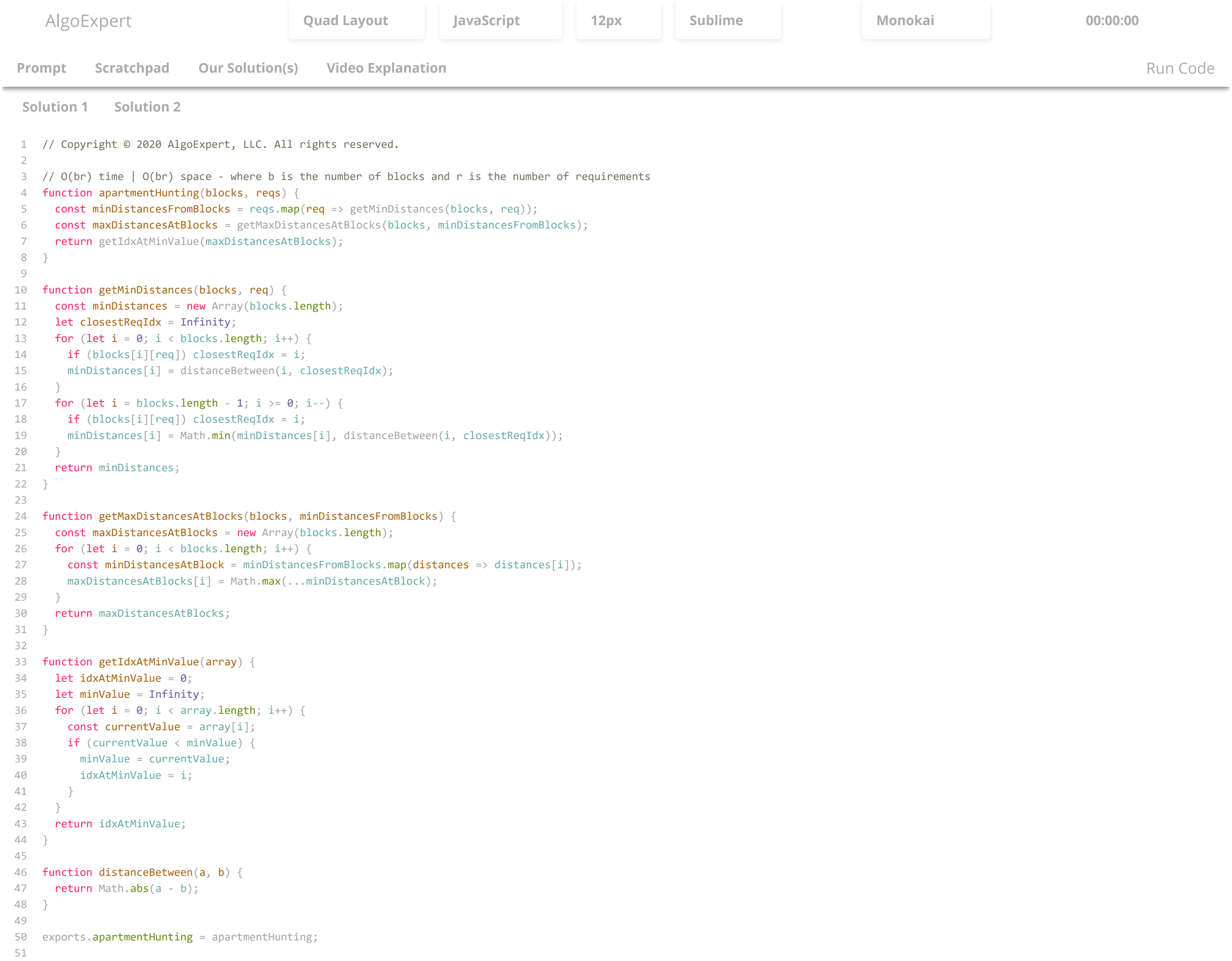

```javascript
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

// O(br) time | O(br) space - where b is the number of blocks and r is the number of requirements
function apartmentHunting(blocks, reqs) {
  const minDistancesFromBlocks = reqs.map(req => getMinDistances(blocks, req));
  const maxDistancesAtBlocks = getMaxDistancesAtBlocks(blocks, minDistancesFromBlocks);
  return getIdxAtMinValue(maxDistancesAtBlocks);
}

function getMinDistances(blocks, req) {
  const minDistances = new Array(blocks.length);
  let closestReqIdx = Infinity;
  for (let i = 0; i < blocks.length; i++) {
    if (blocks[i][req]) closestReqIdx = i;
    minDistances[i] = distanceBetween(i, closestReqIdx);
  }
  for (let i = blocks.length - 1; i >= 0; i--) {
    if (blocks[i][req]) closestReqIdx = i;
    minDistances[i] = Math.min(minDistances[i], distanceBetween(i, closestReqIdx));
  }
  return minDistances;
}

function getMaxDistancesAtBlocks(blocks, minDistancesFromBlocks) {
  const maxDistancesAtBlocks = new Array(blocks.length);
  for (let i = 0; i < blocks.length; i++) {
    const minDistancesAtBlock = minDistancesFromBlocks.map(distances => distances[i]);
    maxDistancesAtBlocks[i] = Math.max(...minDistancesAtBlock);
  }
  return 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;

```