AlgoExpert Quad Layout C# 12px Sublime Monokai 00:00:00

Prompt Scratchpad Our Solution(s) Video Explanation Run Code

Solution 1 Solution 2

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

using System;
using System.Collections.Generic;


public class Program {
  // O(br) time | O(br) space - where b is the number of blocks and r is the number of requirements
  public static int ApartmentHunting(List<Dictionary<string, bool> > blocks, string[] reqs) {
    int[][] minDistancesFromBlocks = new int[reqs.Length][];
    for (int i = 0; i < reqs.Length; i++) {
      minDistancesFromBlocks[i] = getMinDistances(blocks, reqs[i]);
    }
    int[] maxDistancesAtBlocks =
      getMaxDistancesAtBlocks(blocks, minDistancesFromBlocks);
    return getIdxAtMinValue(maxDistancesAtBlocks);
  }

  public static int[] getMinDistances(List<Dictionary<string, bool> > blocks, string req) {
    int[] minDistances = new int[blocks.Count];
    int closestReqIdx = Int32.MaxValue;
    for (int i = 0; i < blocks.Count; i++) {
      if (blocks[i][req]) closestReqIdx = i;
      minDistances[i] = distanceBetween(i, closestReqIdx);
    }
    for (int i = blocks.Count - 1; i >= 0; i--) {
      if (blocks[i][req]) closestReqIdx = i;
      minDistances[i] = Math.Min(minDistances[i], distanceBetween(i,
          closestReqIdx));
    }
    return minDistances;
  }

  public static int[] getMaxDistancesAtBlocks(List<Dictionary<string, bool> > blocks,
    int[][] minDistancesFromBlocks) {
    int[] maxDistancesAtBlocks = new int[blocks.Count];
    for (int i = 0; i < blocks.Count; i++) {
      int[] minDistancesAtBlock = new int[minDistancesFromBlocks.Length];
      for (int j = 0; j < minDistancesFromBlocks.Length; j++) {
        minDistancesAtBlock[j] = minDistancesFromBlocks[j][i];
      }
      maxDistancesAtBlocks[i] = arrayMax(minDistancesAtBlock);
    }
    return maxDistancesAtBlocks;
  }

  public static int getIdxAtMinValue(int[] array) {
    int idxAtMinValue = 0;
    int minValue = Int32.MaxValue;
    for (int i = 0; i < array.Length; i++) {
      int currentValue = array[i];
      if (currentValue < minValue) {
        minValue = currentValue;
        idxAtMinValue = i;
      }
    }
    return idxAtMinValue;
  }

  public static int distanceBetween(int a, int b) {
    return Math.Abs(a - b);
  }

  public static int arrayMax(int[] array) {
    int max = array[0];
    foreach (int a in array) {
      if (a > max) {
        max = a;
      }
    }
    return max;
  }
}
```