**5 kyu**

**Diagonals**

8192% of 1318[surtich](https://www.codewars.com/users/surtich)

JavaScript

* [TRAIN AGAIN](https://www.codewars.com/kata/diagonals/train/javascript)
* [NEXT KATA](https://www.codewars.com/trainer/javascript)

Details

[Solutions](https://www.codewars.com/kata/diagonals/solutions/javascript)

[Discourse (12)](https://www.codewars.com/kata/diagonals/discuss/javascript)

* Add to Collection
* |
* Share this kata:

Create a function that calculates all possible diagonals of a given (square) matrix.

Matrix = array of n length whose elements are n length arrays of integers.

2x2 example:

diagonals( [

[ 1, 2 ],

[ 3, 4 ]

] );

returns -> [ [ 1 ], [ 2, 3 ], [ 4 ], [ 2 ], [ 1, 4 ], [ 3 ] ]

it is valid too -> [ [ 1, 4 ], [ 3 ], [ 2 ], [ 2 , 3 ], [ 1 ], [ 4 ] ] //Order of the returned array does not matter

it is invalid -> [ [ 1 ], [ 3, 2 ], [ 4 ], [ 2 ], [ 1, 4 ], [ 3 ] ] //Order of each diagonal must be preserved

3x3 example:

diagonals( [

[ 1, 2, 3 ],

[ 4, 5, 6 ],

[ 7, 8, 9 ]

] );

returns ->

[ [ 1 ],

[ 2, 4 ],

[ 3, 5, 7 ],

[ 6, 8 ],

[ 9 ],

[ 3 ],

[ 2, 6 ],

[ 1, 5, 9 ],

[ 4, 8 ],

[ 7 ] ]

The tests verify that the implementation is efficient (1000x1000 matrix are used in tests).

<https://www.codewars.com/kata/diagonals/javascript>

1. **function** diagonals(matrix) {

4. **var** listas = [];
5. **if** (matrix.length == 1)
6. {
7. *//return matrix[0][0];*
8. **var** aux = [];
9. aux.push(matrix[0][0]);
10. listas.push(aux);
12. }

15. sumas = {};
16. difs = {};
18. **for**(let i =0; i<matrix.length; i++)
19. {
20. **for**(let j =0; j<matrix[0].length; j++)
21. {
22. **var** suma = i + j;


26. **if** (sumas.hasOwnProperty(suma))
27. {
28. sumas[suma].push(matrix[i][j]);
29. }
30. **else**
31. {
32. sumas[suma] = [];
33. sumas[suma].push(matrix[i][j]);
34. }
36. **var** dif = i - j;
38. **if**( difs.hasOwnProperty(dif))
39. {
40. difs[dif].push(matrix[i][j]);
41. }
42. **else**
43. {
44. difs[dif] = [];
45. difs[dif].push(matrix[i][j]);
46. }
48. }
50. }
52. listas = [];
54. **for**(let key **in** sumas)
55. {
56. **var** val = sumas[key];
57. listas.push(val);
58. }

61. **for**(let key **in** difs)
62. {
63. **var** val = difs[keys];
64. listas.push(val);
65. }


69. **return** listas;
71. }

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static List<List<int>> diagonals(int[][] matrix)

{

List<List<int>> listas = new List<List<int>>();

if (matrix.Length == 1)

{

//return matrix[0][0];

List<int> aux = new List<int>();

aux.Add(matrix[0][0]);

listas.Add(aux);

}

Dictionary<int, List<int>> sumas = new Dictionary<int, List<int>>();

Dictionary<int, List<int>> difs = new Dictionary<int, List<int>>();

for (int i = 0; i < matrix.Length; i++)

{

for (int j = 0; j < matrix[0].Length; j++)

{

int suma = i + j;

if (sumas.ContainsKey(suma))

{

sumas[suma].Add(matrix[i][j]);

}

else

{

sumas[suma] = new List<int>();

sumas[suma].Add(matrix[i][j]);

}

int dif = i - j;

if (difs.ContainsKey(dif))

{

difs[dif].Add(matrix[i][j]);

}

else

{

difs[dif] = new List<int>();

difs[dif].Add(matrix[i][j]);

}

}

}

//List<List<int>> listas = new List<List<int>>();

foreach (KeyValuePair<int, List<int>> kvp in sumas)

{

List<int> val = kvp.Value;

//Console.Write(kvp.Key + " : ");

//foreach(int item in val)

//{

// Console.Write(item + " ");

//}

listas.Add(val);

//Console.WriteLine();

}

Console.WriteLine();

foreach (KeyValuePair<int, List<int>> kvp in difs)

{

List<int> val = kvp.Value;

//Console.Write(kvp.Key + " : ");

//foreach (int item in val)

//{

// Console.Write(item + " ");

//}

listas.Add(val);

//Console.WriteLine();

}

//foreach(List<int> lista in listas)

//{

// foreach(int item in lista)

// {

// Console.Write(item + " ");

// }

// Console.WriteLine();

//}

return listas;

}

static void Main(string[] args)

{

int[][] m =

{

new int[] { 1, 2, 3 },

new int[] { 4, 5, 6 },

new int[] { 7, 8, 9 }

};

//int[][] m =

// {

// new int[] { 4 },

// };

////Test.assertSimilar(sort(diagonals([[1, 2, 3], [4, 5, 6], [7, 8, 9]])), [[1],[1,5,9],[2,4],[2,6],[3],[3,5,7],[4,8],[6,8],[7],[9]]);

//diagonals(m);

List<List<int>> listas = diagonals(m);

foreach(List<int> lis in listas)

{

foreach(int item in lis)

{

Console.Write(item + " ");

}

Console.WriteLine();

}

Console.ReadLine();

}

}

}