**5 kyu**

**#9 Matrices: Adding diagonal products**

10494% of 2654 of92[raulbc777](https://www.codewars.com/users/raulbc777)

Python

* [TRAIN AGAIN](https://www.codewars.com/kata/number-9-matrices-adding-diagonal-products/train/python)
* [NEXT KATA](https://www.codewars.com/trainer/python)

Details

[Solutions](https://www.codewars.com/kata/number-9-matrices-adding-diagonal-products/solutions/python)

[Discourse (27)](https://www.codewars.com/kata/number-9-matrices-adding-diagonal-products/discuss/python)

* Add to Collection
* |
* Share this kata:

We have a square matrix M of dimension n x n that has positive and negative numbers in the ranges [-9,-1] and [0,9],( the value 0 is excluded).

We want to add up all the products of the elements of the diagonals UP-LEFT to DOWN-BOTTOM, that is the value ofsum1; and the elements of the diagonals UP-RIGHT to LEFT-DOWN and that is sum2. Then, as a final result, the value of sum1 - sum2.

E.g.

M = [[ 1, 4, 7, 6, 5],

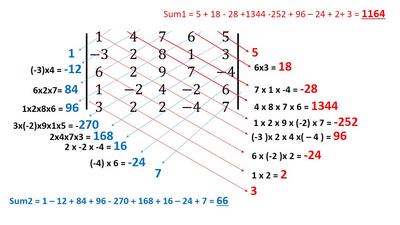
[-3, 2, 8, 1, 3],

[ 6, 2, 9, 7, -4],

[ 1, -2, 4, -2, 6],

[ 3, 2, 2, -4, 7]]

Let's see how to get this result in the image below:

[](http://imgur.com/MHfydrP)

So the value of sum1 - sum2 is equal to:

1164 - 66 = 1098

Create the code to do this calculation.

Features of the random tests:

Numbers of tests = 150

10 < dimension < 25 (python and ruby)

10 < dimension < 20 (javascript)

-10 < M[i,j] < 0 and 0 < M[i,j] < 10

This kata is available in Python2, Ruby and Javascript by the moment. Translations into another languages will be released soon. Enjoy it!

<https://www.codewars.com/kata/number-9-matrices-adding-diagonal-products/python>

static int sum\_prod\_diags(int[][] matrix)

{

int n = matrix.Length;

int[] psum = new int[2 \* n - 1];

for (int i = 0; i < psum.Length; i++) psum[i] = 1;

Dictionary<int, int> pdif = new Dictionary<int, int>();

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

{

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

{

int dif = i - j;

//pdif[ dif ] =

if (pdif.ContainsKey(dif))

{

pdif[dif] \*= matrix[i][j];

}

else

{

pdif[dif] = matrix[i][j];

}

psum[i + j] \*= matrix[i][j];

}

}

int a = 0;

for (int i = 0; i < psum.Length; i++) a += psum[i];

int b = 0;

foreach (KeyValuePair<int, int> kvp in pdif) b += kvp.Value;

return b - a;

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

//static int sum\_prod\_diags(int[][] matrix)

//{

// int n = matrix.Length;

// int sum2 = 0;

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

// for (int i = n - 1; i >= 0; i--)

// {

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

// int f = i;

// int c = n - 1;

// int prod2 = 1;

// while (f < n && c >= 0)

// {

// // Console.Write(matrix[f][c] + " ");

// //fila.Add(matrix[f][c]);

// prod2 \*= matrix[f][c];

// f++;

// c--;

// }

// sum2 += prod2;

// // ans.Add(fila);

// }

// for (int j = n - 2; j >= 0; j--)

// {

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

// int f = 0, c = j;

// int prod2 = 1;

// while (f < n && c >= 0)

// {

// //Console.Write(matrix[f][c] + " ");

// //fila.Add(matrix[f][c]);

// prod2 \*= matrix[f][c];

// f++;

// c--;

// }

// sum2 += prod2;

// }

// int sum1 = 0;

// for (int j = n - 1; j >= 0; j--)

// {

// int f = 0, c = j;

// int prod1 = 1;

// while (f < n && c < n)

// {

// //Console.Write(matrix[f][c] + " ");

// prod1 \*= matrix[f][c];

// f++;

// c++;

// }

// sum1 += prod1;

// }

// for(int i = 1; i<n; i++)

// {

// int f = i, c = 0;

// int prod1 = 1;

// while (f < n && c < n)

// {

// //Console.Write(matrix[f][c] + " ");

// prod1 \*= matrix[f][c];

// f++;

// c++;

// }

// sum1 += prod1;

// }

// return sum1-sum2;

//}

static int sum\_prod\_diags(int[][] matrix)

{

int n = matrix.Length;

int sum2 = 0;

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

for (int i = n - 1; i >= 0; i--)

{

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

int f = i;

int c = n - 1;

int prod2 = 1;

while (f < n && c >= 0)

{

// Console.Write(matrix[f][c] + " ");

//fila.Add(matrix[f][c]);

prod2 \*= matrix[f][c];

f++;

c--;

}

sum2 += prod2;

// ans.Add(fila);

}

for (int j = n - 2; j >= 0; j--)

{

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

int f = 0, c = j;

int prod2 = 1;

while (f < n && c >= 0)

{

//Console.Write(matrix[f][c] + " ");

//fila.Add(matrix[f][c]);

prod2 \*= matrix[f][c];

f++;

c--;

}

sum2 += prod2;

}

int sum1 = 0;

for (int j = n - 1; j >= 0; j--)

{

int f = 0, c = j;

int prod1 = 1;

while (f < n && c < n)

{

//Console.Write(matrix[f][c] + " ");

prod1 \*= matrix[f][c];

f++;

c++;

}

sum1 += prod1;

}

for (int i = 1; i < n; i++)

{

int f = i, c = 0;

int prod1 = 1;

while (f < n && c < n)

{

//Console.Write(matrix[f][c] + " ");

prod1 \*= matrix[f][c];

f++;

c++;

}

sum1 += prod1;

}

return sum1 - sum2;

}

static void Main(string[] args)

{

int[][] m =

{

new int[]{1,4,7,6,5},

new int[]{-3,2,8,1,3},

new int[]{6,2,9,7,-4},

new int[]{ 1,-2,4,-2,6 },

new int[]{ 3,2,2,-4,7}

};

Console.WriteLine( sum\_prod\_diags(m));

/\*

foreach(List<int> lista in ans)

{

foreach(int item in lista)

{

Console.Write(item + " ");

}

Console.WriteLine();

}

\*/

Console.ReadLine();

}

}

}

**def** sum\_prod\_diags(matrix):

    n = len(matrix)

*#ans = []*

    sum2 = 0

**for** i **in** range(n-1, -1, -1):

*#fila = []*

        f = i

        c = n - 1

        prod2 = 1

**while**( f < n **and** c >=0):

*#ans.append(matrix[f][c])*

            prod2 \*= matrix[f][c]

            f+=1

            c-=1

        sum2 += prod2

**for** j **in** range(n-2, -1, -1):

*#fila = []*

        f = 0

        c = j

        prod2 = 1

**while**(f < n **and** c >=0):

*#ans.append(matrix[f][c])*

            prod2 \*= matrix[f][c]

            f += 1

            c -= 1

        sum2 += prod2

    sum1 = 0

**for** j **in** range(n-1, -1, -1 ):

        f = 0

        c = j

        prod1 = 1

**while**(f < n **and** c < n):

            prod1 \*= matrix[f][c]

            f+=1

            c+=1

        sum1 += prod1

**for** i **in** range(1, n):

        f = i

        c = 0

        prod1 = 1

**while**(f < n **and** c < n):

            prod1 \*= matrix[f][c]

            f+=1

            c+=1

        sum1 += prod1

**return** sum1 - sum2

M1 = [[ 1,  4, 7,  6,  5],

     [-3,  2, 8,  1,  3],

     [ 6,  2, 9,  7, -4],

     [ 1, -2, 4, -2,  6],

     [ 3,  2, 2, -4,  7]]

**print**(diagonal(M1))

**def** sum\_prod\_diags(matrix):

    n = len(matrix)

    psum =[1] \* (2\*n-1)

    pdif = {}

    n = len(matrix)

**for** i **in** range(0, n ):

**for** j **in** range(0, len(matrix[i])):

            dif = i-j

**if** (dif **in** pdif):

                pdif[dif] \*= matrix[i][j]

**else**:

                pdif[dif] = matrix[i][j]

            psum[i+j] \*= matrix[i][j]

    a = 0

**for** i **in** range(0, len(psum)): a += psum[i]

    b = 0

**for** key **in** pdif:

        b += pdif[key]

**return** b-a