|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 2 | 0 | 1 |
| 1 | 2 | 1 | 1 |
| 1 | 3 | 1 | 1 |
| 2 | 3 | 1 | 1 |

P Q result

|  |
| --- |
| 2 |
| 5 |
| 0 |

|  |
| --- |
| 4 |
| 5 |
| 6 |

Nucleo[Q[count]+1 , 0] – nucleo[P[count] , 0)

Buc(5,0) – nuv (2,0) 1-1 = 0

Nuc(5,1) – nuc(2,1) 2-1 = 1

True

|  |
| --- |
| 2 |
| 4 |
| 1 |

Lesson 5:

Excercise 1 : CountDiv

<https://app.codility.com/demo/results/trainingSAMWD4-S5E/>

using System;

// you can also use other imports, for example:

// using System.Collections.Generic;

// you can write to stdout for debugging purposes, e.g.

// Console.WriteLine("this is a debug message");

class Solution {

public int solution(int A, int B, int K) {

while (A % K != 0) {

++A;

}

while (B % K != 0) {

--B;

}

return (B - A) / K + 1;

}

}

// write your code in C# 6.0 with .NET 4.5 (Mono)

Exercise 2:

GenomicRangeQuery

<https://app.codility.com/demo/results/training4S4Q6U-H4H/>

using System;

// you can also use other imports, for example:

// using System.Collections.Generic;

// you can write to stdout for debugging purposes, e.g.

// Console.WriteLine("this is a debug message");

class Solution {

public int[] solution(string S, int[] P, int[] Q) {

var nucleo = new int[S.Length + 1, 4];

for (var count = 0; count < S.Length; count++)

{

if (count > 0)

{

for (var index = 0; index < 4; index++)

{

nucleo[count + 1, index] += nucleo[count, index];

}

}

switch (S[count])

{

case 'A':

nucleo[count + 1, 0]++;

break;

case 'C':

nucleo[count + 1, 1]++;

break;

case 'G':

nucleo[count + 1, 2]++;

break;

case 'T':

nucleo[count + 1, 3]++;

break;

}

}

var result = new int[P.Length];

for (var count = 0; count < P.Length; count++) {

if(P[count] == Q[count])

{

switch(S[P[count]]) {

case 'A':

result[count] = 1;

break;

case 'C':

result[count] = 2;

break;

case 'G':

result[count] = 3;

break;

case 'T':

result[count] = 4;

break;

}

} else {

for(var index = 0; index < 4; index++) {

if((nucleo[Q[count] + 1, index] - nucleo[P[count], index]) > 0) {

result[count] = index + 1;

break;

}

}

}

}

return result;

// write your code in C# 6.0 with .NET 4.5 (Mono)

}

}

Exercise 3:

MinAvgTwoSlice

<https://app.codility.com/demo/results/training3W2DAE-8HC/>

using System;

// you can also use other imports, for example:

// using System.Collections.Generic;

// you can write to stdout for debugging purposes, e.g.

// Console.WriteLine("this is a debug message");

class Solution {

public int solution(int[] A) {

int minI=0;

double minValue = 100001.0;

for (int i=0; i<A.Length-1; i++)

{

if (((A[i]+A[i+1])/2.0) < minValue)

{

minI=i;

minValue=(A[i]+A[i+1])/2.0;

}

if (i < A.Length-2)

{

if (((A[i] +A[i+1]+A[i+2])/3.0)< minValue)

{

minI=i;

minValue= (A[i] +A[i+1]+A[i+2]) / 3.0;

}

}

}

return minI;

// write your code in C# 6.0 with .NET 4.5 (Mono)

}

}

------------------------------------------------------------------------------------------------

Exercise 4:

PassingCars

using System;

// you can also use other imports, for example:

// using System.Collections.Generic;

// you can write to stdout for debugging purposes, e.g.

// Console.WriteLine("this is a debug message");

class Solution {

public int solution(int[] A) {

int l = A.Length;

int countZero =0;

int countPairs=0;

int exceed = 1000000000;

for (int i=0; i<l; i++)

{

if (A[i]==1) countPairs += countZero;

else countZero++;

}

if (countPairs > exceed || countPairs <0) return -1;

else return countPairs;

}

// write your code in C# 6.0 with .NET 4.5 (Mono)

}