<https://www.hackerrank.com/contests/womens-codesprint-5/challenges/recycled-number/problem>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp92

{

class Program

{

public static String obtenerMenor(String s)

{

String concat = s + s;

SortedSet<String> sortedSet =

new SortedSet<string>();

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

{

sortedSet.Add(concat.Substring(i, s.Length));

}

return sortedSet.First();

}

static long uniqueRecycledPairs(int[] A)

{

Dictionary<string, HashSet<int>> dictionary = new Dictionary<string, HashSet<int>>();

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

{

string menor = obtenerMenor(A[i].ToString());

if (dictionary.ContainsKey(menor))

{

dictionary[menor].Add(A[i]);

}

else

{

HashSet<int> ss = new HashSet<int>(); //no hace falta que sea un sortedset

ss.Add(A[i]);

dictionary[menor] = ss;

}

}

long ans = 0;

foreach (KeyValuePair<string, HashSet<int>> kvp in dictionary)

{

HashSet<int> aux = kvp.Value;

int n = aux.Count;

if (n >= 2)

{

ans += (n \* (n - 1)) / 2;

}

}

return ans;

}

static void Main(string[] args)

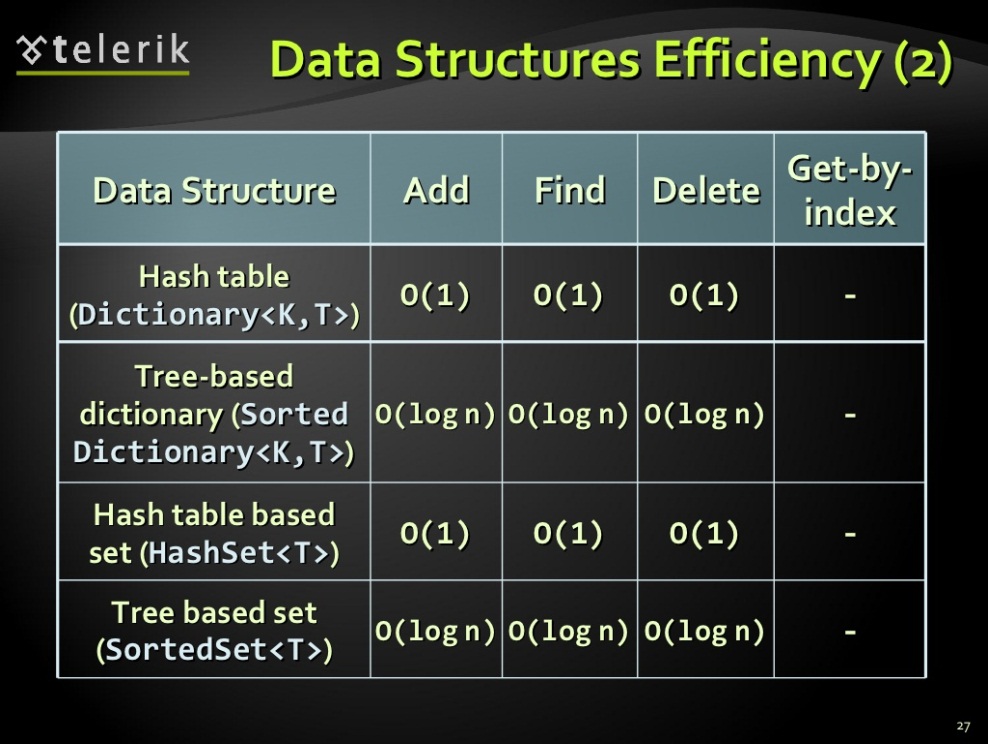
{

}

}

}

Con diccionario de sortedset: Dictionary<string, SortedSet<int>> dictionary = new Dictionary<string, SortedSet<int>>();



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp85

{

class Program

{

static bool contiene(string seg, string prim)

{

if (seg.Length != prim.Length) return false;

return (seg + seg).Contains(prim);

}

public static String obtenerMenor(String s)

{

String concat = s + s;

SortedSet<String> sortedSet =

new SortedSet<string>();

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

{

sortedSet.Add(concat.Substring(i, s.Length));

}

return sortedSet.First();

}

static long uniqueRecycledPairs(int[] A)

{

Dictionary<string, SortedSet<int>> dictionary = new Dictionary<string, SortedSet<int>>();

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

{

string menor = obtenerMenor(A[i].ToString());

if (dictionary.ContainsKey(menor))

{

dictionary[menor].Add(A[i]);

}

else

{

SortedSet<int> ss = new SortedSet<int>();

ss.Add(A[i]);

dictionary[menor] = ss;

}

}

//foreach(KeyValuePair<string, SortedSet<int>> kvp in dictionary)

//{

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

// SortedSet<int> aux = kvp.Value;

// foreach(int elem in aux)

// {

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

// }

// Console.WriteLine();

//}

long ans = 0;

foreach (KeyValuePair<string, SortedSet<int>> kvp in dictionary)

{

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

//SortedSet<int> aux = kvp.Value;

//foreach (int elem in aux)

//{

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

//}

//Console.WriteLine();

SortedSet<int> aux = kvp.Value;

int n = aux.Count;

if(n >= 2)

{

ans += (n \* (n - 1)) / 2;

}

}

return ans;

}

static void Main(string[] args)

{

int seg = 3412, prim = 1234;

//34123412

//12341234

// Console.WriteLine(contiene(prim.ToString(), seg.ToString()));

//string[] nums = { "1432", "8973", "90" };

//Array.Sort(nums);

//foreach(string elem in nums)

//{

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

//}

int[] a = { 10, 1, 13, 31, 23, 32 };

Console.WriteLine( uniqueRecycledPairs(a));

Console.ReadLine();

}

}

}