**String Matching**

Attempted by: **1188**

/

Accuracy: **75%**

/

Maximum Score: **30**

/

14 Votes

Tag(s):

Algorithms, Dynamic Programming, Easy-Medium

**PROBLEM**

**EDITORIAL**

**MY SUBMISSIONS**

**ANALYTICS**

Given a string XX formed out of single digit numbers from 0−90−9 , you are given a set of digits SS and you need to count total substring of string XX that contains all the digits in the set SS.   
**Input**   
First line contains a string as input. Next line contains a number nn as input denoting size of set SS. Next line contains nnspace separated integers that denote the distinct integers in the set SS.   
**Output**  
In the output you have to give total count of substrings of the string XX such that they contain all the digits in the set SS  
**Constraints**  
1≤|X|≤1051≤|X|≤105  
1≤n≤101≤n≤10

**SAMPLE INPUT**

333

1

3

**SAMPLE OUTPUT**

6

**Explanation**

There are 6 substrings that have 3 in them

**Time Limit:**1.0 sec(s) for each input file.

**Memory Limit:**1024 MB

**Source Limit:**1024 KB

**Marking Scheme:**Marks are awarded when all the testcases pass.

**Allowed Languages:**C, C++, C++14, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java, Java 8, JavaScript(Rhino), JavaScript(Node.js), Julia, Kotlin, Lisp, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, R(RScript), Racket, Ruby, Rust, Scala, Swift, Visual Basic

<https://www.hackerearth.com/practice/algorithms/dynamic-programming/bit-masking/practice-problems/algorithm/string-matching-google-3dc355a5/>

[String Matching](https://www.hackerearth.com/practice/algorithms/dynamic-programming/bit-masking/practice-problems/algorithm/string-matching-google-3dc355a5/) / [Submission (10889679)](https://www.hackerearth.com/challenge/competitive/august-circuits-17/algorithm/string-matching-google-3dc355a5/submission/10889679/) by [Tinkesh Kumar (sumitb107nitw)](https://www.hackerearth.com/@sumitb107nitw)

using System;

using System.Collections.Generic;

using System.Linq;

class MyClass

{

static void Main(string[] args)

{

string stringinput = Console.ReadLine().Trim();

Console.ReadLine();

var subset = Console.ReadLine().Trim().Split(' ').Select(z => int.Parse(z)).ToList();

var outputcount = GetCount(stringinput, subset);

Console.WriteLine(outputcount);

}

public static long GetCount(string input, List<int> subset)

{

long count = 0;

int inputlength = input.Length;

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

{

var tempSubSet = subset.ToDictionary(x => x, x => x);

int k = 0;

int tempCount = 0;

for (int j = i; j < inputlength; j++)

{

if (tempSubSet.ContainsKey(input[j] - '0'))

{

tempSubSet.Remove(input[j] - '0');

}

if (tempSubSet.Count == 0)

{

k = j;

break;

}

}

if (tempSubSet.Count > 0)

break;

if (k >= 0)

{

tempCount = inputlength - k;

}

tempSubSet = subset.ToDictionary(x => x, x => x);

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

{

count = count + tempCount;

if (tempSubSet.ContainsKey(input[l] - '0'))

{

break;

}

i++;

}

}

return count;

}

}

[String Matching](https://www.hackerearth.com/practice/algorithms/dynamic-programming/bit-masking/practice-problems/algorithm/string-matching-google-3dc355a5/) / [Submission (10965033)](https://www.hackerearth.com/challenge/competitive/august-circuits-17/algorithm/string-matching-google-3dc355a5/submission/10965033/) by [Satyendra Kumar Bilthare (biltharesatyendra)](https://www.hackerearth.com/@biltharesatyendra)

using System;

using System.Collections.Generic;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

// (づ°ω°)づﾐe★゜・。。・゜゜・。。・゜☆゜・。。・゜゜・。。・゜

public class Solver

{

public void Solve()

{

string s = ReadToken();

var f = new bool[128];

int m = ReadInt();

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

f[ReadToken()[0]] = true;

long ans = 0;

int p = 0;

int n = s.Length;

int cnt = 0;

var c = new int[128];

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

{

while (p < n && cnt < m)

{

if (f[s[p]])

{

if (c[s[p]]++ == 0)

cnt++;

}

p++;

}

if (cnt == m)

ans += n - p + 1;

if (f[s[i]] && c[s[i]]-- == 1)

cnt--;

}

Write(ans);

}

#region Main

protected static TextReader reader;

protected static TextWriter writer;

static void Main()

{

#if DEBUG

reader = new StreamReader("..\\..\\input.txt");

//reader = new StreamReader(Console.OpenStandardInput());

writer = Console.Out;

//writer = new StreamWriter("..\\..\\output.txt");

#else

reader = new StreamReader(Console.OpenStandardInput());

writer = new StreamWriter(Console.OpenStandardOutput());

//reader = new StreamReader("input.txt");

//writer = new StreamWriter("output.txt");

#endif

try

{

new Solver().Solve();

//var thread = new Thread(new Solver().Solve, 1024 \* 1024 \* 128);

//thread.Start();

//thread.Join();

}

catch (Exception ex)

{

#if DEBUG

Console.WriteLine(ex);

#else

throw;

#endif

}

reader.Close();

writer.Close();

}

#endregion

#region Read / Write

private static Queue<string> currentLineTokens = new Queue<string>();

private static string[] ReadAndSplitLine() { return reader.ReadLine().Split(new[] { ' ', '\t', }, StringSplitOptions.RemoveEmptyEntries); }

public static string ReadToken() { while (currentLineTokens.Count == 0)currentLineTokens = new Queue<string>(ReadAndSplitLine()); return currentLineTokens.Dequeue(); }

public static int ReadInt() { return int.Parse(ReadToken()); }

public static long ReadLong() { return long.Parse(ReadToken()); }

public static double ReadDouble() { return double.Parse(ReadToken(), CultureInfo.InvariantCulture); }

public static int[] ReadIntArray() { return ReadAndSplitLine().Select(int.Parse).ToArray(); }

public static long[] ReadLongArray() { return ReadAndSplitLine().Select(long.Parse).ToArray(); }

public static double[] ReadDoubleArray() { return ReadAndSplitLine().Select(s => double.Parse(s, CultureInfo.InvariantCulture)).ToArray(); }

public static int[][] ReadIntMatrix(int numberOfRows) { int[][] matrix = new int[numberOfRows][]; for (int i = 0; i < numberOfRows; i++)matrix[i] = ReadIntArray(); return matrix; }

public static int[][] ReadAndTransposeIntMatrix(int numberOfRows)

{

int[][] matrix = ReadIntMatrix(numberOfRows); int[][] ret = new int[matrix[0].Length][];

for (int i = 0; i < ret.Length; i++) { ret[i] = new int[numberOfRows]; for (int j = 0; j < numberOfRows; j++)ret[i][j] = matrix[j][i]; } return ret;

}

public static string[] ReadLines(int quantity) { string[] lines = new string[quantity]; for (int i = 0; i < quantity; i++)lines[i] = reader.ReadLine().Trim(); return lines; }

public static void WriteArray<T>(IEnumerable<T> array) { writer.WriteLine(string.Join(" ", array)); }

public static void Write(params object[] array) { WriteArray(array); }

public static void WriteLines<T>(IEnumerable<T> array) { foreach (var a in array)writer.WriteLine(a); }

private class SDictionary<TKey, TValue> : Dictionary<TKey, TValue>

{

public new TValue this[TKey key]

{

get { return ContainsKey(key) ? base[key] : default(TValue); }

set { base[key] = value; }

}

}

private static T[] Init<T>(int size) where T : new() { var ret = new T[size]; for (int i = 0; i < size; i++)ret[i] = new T(); return ret; }

#endregion

}