Реализовать метод, который принимает на вход строку и возвращает коллекцию символов, которые встречаются чаще всего

Пример:

- aaa bbb c

- a, b

- aaa bb c

- a

- a, b, c

- a b c

Как модифицировать алгоритм, чтобы количество итераций было меньше, чем символов в строке.

Привет, вот ответ. Извините, что на бейсике, я на нем пишу намного быстрее так как ок 20-ти лет пишу на нем. Впрочем, сейчас все конвертеры http://converter.telerik.com/ конвертируют бейсик в шарп без проблем (ну если конечно не пользоваться многими могучими возможностями бейсика, отсуствующими в шарпе, а лишь тем небольшим подмножеством функционала бейсика, аналоги которому существуют аналог в шарпе).

Итак, есть много путей решения этой задачки. Во-первых, тут можно повозиться ручками с массивами, допустим создать два одномерных массива, в одном сами символы, во втором их количество, просто проходим простым циклом по массиву и проверяем количество. Уменьшить количество итераций можно с помощью третьего массива, в котором будет отмечаться флаг, что по этому символу уже проверка количества его вложений прошла. Таким образом во внешнем цикле будет итераций всегда меньше, чем символов в строке. Какая-то очень нудная возня с массивами, просто два цикла один внутри другого, задачка на вложенные циклы...

Мне показалось более интересным решение с использованием именно NET-фреймворк, это ж задачка не просто по вложенным циклам в абстрактной среде, а именно в NET? А тут есть миллион возможностей для красивых решений вообще без циклов. Точнее циклы тут присутствуют в неявном виде, утопленные в MSIL. Вот одно из таких интересных решений. В нем сначала проводится группировка символов средствами самого NET. Вот эта функция в строчках 24-34 создается по одному нажатию Enter, ничего ручками там писать не надо. Особенно в Бейсике (это в Шарпе надо ставить Решарпер, и он все равно лишь малую часть реализует функций, которые уже встроены в ядро любой Visual Studio для поддержки бейсика, для бейсика же в любой версии студии это фактически один тык в Enter).

Внешних циклов в этом решении будет ровно столько, сколько повторяющихся символов будет уже после группировки, то есть если строка состоит из двух символов, то внешних проходов в таком подходе будет только два, чтобы определить, каких элементов больше. Можно избавиться и от этого небольшого внешнего цикла (стр 14-19), записав вместо него конструкцию OrderBy - то есть в этом случае в коде высокого уровня циклов не будет вообще. Первый вызов Group, второй Order. Внутри этих циклов, на уровне MSIL - циклы конечно будут, но в коде высокого уровня таких циклов нет, будет просто два оператора NET-фреймворк, которые и решают требуемую задачку. Правда второй оператор получается сильно замороченный и непонятно как из него вытащить результат для возврата из функции. Поэтому, для простоты я развернул второй оператор OrderBy в виде цикла. Выполняться он будет уже по сгруппированным символам, то есть гарантированно меньше раз, чем количество символов в строке, вернуть нужный результат в этом решении легко, условия задачки выполнены, поэтому я и остановился на этом варианте код.

Как видите, решение заняло всего 30 строчек, из которых со строчки 24 студия создала сама (шаблон интерфейса Icomparer), а до 10-й строчки - это демо-вызов функции. Сам же смысловой код занимает всего 10 строчек, с 10-й по 20-ю, отвечает всем требованиям задачки, возвращает ровно то что требуется и отвечает условию задачки - количество итераций в коде высокого уровня гарантированно меньше, чем количество символов в строке.

С уважением, Вячеслав (https://www.linkedin.com/in/viacheslav-eremin-59b459b2)

```
Module Module1


    Sub Main()

        Console.WriteLine(String.Join("", Xfunc(" aaa bbb c, ddd
,eeec,,,cseree[kmjesdfsfeeeee,e").ToArray))

        While True

            Console.WriteLine(String.Join("", Xfunc(Console.ReadLine()).ToArray))

        End While

    End Sub


    Function Xfunc(Str1 As String) As IList(Of Char)

        Dim Arr1 As Char() = Str1.ToArray

        Dim Z = (Arr1.GroupBy(Function(Y) Y, New MyCharComparer())).ToList()

        Dim Count As Integer = 0, Ind As Integer = 0

        For i As Integer = 0 To Z.Count - 1

            If Z(i).ToList.Count > Count Then

                Count = Z(i).Count

                Ind = i

            End If

        Next

        Return Z(Ind).ToList()

    End Function

End Module


Public Class MyCharComparer

    Implements IEqualityComparer(Of Char)


    Public Shadows Function Equals1(x As Char, y As Char) As Boolean Implements
System.Collections.Generic.IEqualityComparer(Of Char).Equals

        Return x = y

    End Function


    Public Shadows Function GetHashCode1(obj As Char) As Integer Implements
System.Collections.Generic.IEqualityComparer(Of Char).GetHashCode

        Return obj.GetHashCode

    End Function

End Class
```

Module1.vb

Module1 | Main

```
Module Module1

    Sub Main()
        Console.WriteLine(String.Join("", Xfunc(" aaa bbb c, ddd ,eeec,,,cseree[kmjesdfsfeeeee,e").ToArray))
        While True
            Console.WriteLine(String.Join("", Xfunc(Console.ReadLine()).ToArray))
        End While
    End Sub

    Function Xfunc(Str1 As String) As IList(Of Char)
        Dim Arr1 As Char() = Str1.ToArray
        Dim Z = (Arr1.GroupBy(Function(Y) Y, New MyCharComparer())).ToList()
        Dim Count As Integer = 0, Ind As Integer = 0
        For i As Integer = 0 To Z.Count - 1
            If Z(i).ToList.Count > Count Then
                Count = Z(i).Count
                Ind = i
            End If
        Next
        Return Z(Ind).ToList()
    End Function
End Module

Public Class MyCharComparer
    Implements IEqualityComparer(Of Char)

    Public Shadows Function Equals1(x As Char, y As Char) As Boolean Implements System.Collections.Generic.IEqualityComparer(Of Char).Equals
        Return x = y
    End Function

    Public Shadows Function GetHashCode1(obj As Char) As Integer Implements System.Collections.Generic.IEqualityComparer(Of Char).GetHashCode
        Return obj.GetHashCode
    End Function
End Class
```

file:///G:/Projects/All_Test/ConsoleApplication3/ConsoleApplication3/bin/Debug/ConsoleApplication...

```
eeeeeeeeeeeee
bhbppoqwezxcnmsdfbhgfstyafsbnvqadnvzxcytsfc
bbbb
yhvhbsdhjbfnmxcbsdgvfewgrhjwebfnmjczcxxz
hhhh
888888888888888schjbsghcvbbhc
888888888888888
ncbdsbvfghrvefb xcnb xznbc nasdvewrabxanasbvdhqwvgd
bbbbbbb
ojrfkcbxmcbxzmbcmzxbcxzjbcyghrmnbcmzbxnc
ccccccc
```

100 %