```
public static class GetContainedNumbersClass
    {
        public static IEnumerable<int> GetContainedNumbers(this IEnumerable<string>?
             source)
        {
            IEnumerable<int> GetContainedNumbers_Aux()
            {
                using (var s = source.GetEnumerator())
                {
                    while (s.MoveNext())
                    {
                        var sb = new StringBuilder();
                        foreach (var c in s.Current)
                        {
                            if(Char.IsDigit(c)) sb.Append(c);
                        }
                        var result = sb.ToString();
                        if ("" == result)
                          throw new ArgumentException("una stringa non contiene
                                numeri");
                        yield return Int32.Parse(result);
                    }
                }
            }
            if(null == source)
               throw new ArgumentNullException(nameof(source), "la sequenza non
                    può essere nulla");
            return GetContainedNumbers_Aux();
        }
    }

    public class GetContainedNumbersTest
    {
        [Test]
        public void NoNumberInSequence() =>
            Assert.That(() => new[] { "45", "-8gg", "||| sembra ma non lo e'"
             }.GetContainedNumbers().ToArray(), Throws.TypeOf<ArgumentException>());
        [Test]
        public void NormalBehaviour() =>
            Assert.That(new[] { "f55h7", "90", "-45", "H 6YY5"
             }.GetContainedNumbers().ToArray(), Is.EqualTo(new[] { 557, 90, 45, 65
             }));
        [Test]
        public void InfiniteSequenceTest()
        {
            IEnumerable<string> Infinite()
            {
                int i = 0;
                while (true)
                {
                    yield return "a" + (i++) + "z";
                }
            }

            var ris = new int[100];

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


            Assert.That(Infinite().Take(100).GetContainedNumbers(), Is.EqualTo(ris));
        }
    }
```