```
public static class MinUpToNowClass
{
    public static IEnumerable<T> MinUpToNow<T>(this IEnumerable<T> leftSeq,
         IEnumerable<T> rightSeq) where T : IComparable<T>
    {
        static IEnumerable<T> MinUpToNow_Aux(IEnumerable<T> leftSeq,
         IEnumerable<T> rightSeq)
        {
            using var it1 = leftSeq.GetEnumerator();
            using var it2 = rightSeq.GetEnumerator();
            bool check1, check2;
            T? minElem = default;
            while (true)
            {
                check1 = it1.MoveNext();
                check2 = it2.MoveNext();

                if (null == minElem) minElem = it1.Current;
                if (check1 == false && check2 == false) break;
                if((check1 == false && check2) || (check1 && check2 == false))
                    throw new ArgumentException("le sequenze hanno lunghezza
                     diversa");
                if (null == it1.Current || null == it2.Current)
                     throw new ArgumentNullException("elemento nullo nella
                     sequenza");
                if (it1.Current.CompareTo(minElem) <= 0)
                {
                    minElem = it1.Current;
                }
                if (it2.Current.CompareTo(minElem) <= 0)
                {
                    minElem = it2.Current;
                }

                yield return minElem;
            }
        }

        if (leftSeq == null || rightSeq == null)
               throw new ArgumentNullException("sequenza nulla");

        return MinUpToNow_Aux(leftSeq, rightSeq);
    }
}

public class MinUpToTopTest
{
    [Test]
    public void DifferentLenght()
    {
        IEnumerable<string> InfiniteSeq()
        {
            while (true) yield return "stringa";
        }
        Assert.That(() => (new[] { "qui", "quo", "qua"
         }.MinUpToNow(InfiniteSeq().Take(20)).ToArray()),
         Throws.InstanceOf<ArgumentException>());
    }
```

```csharp
        [Test]
        public void NormalBehaviour() =>
            Assert.That(new[] { "qui", "quo", "qua", "paperino", "paperone" }
            .MinUpToNow(new[] { "topolino", "pippo", "pluto", "tip", "tap" }),
                Is.EqualTo(new[] { "qui", "pippo", "pippo", "paperino", "paperino"
             }));

        [TestCase(1)]
        public void ErrorIndex(int errorIndex)
        {
            IEnumerable<string> InfiniteSeqRosa(){ while (true) yield return "rosa"; }
            IEnumerable<string> InfiniteSeqViola(){
                int count = 0;
                while (true){
                    if (count == errorIndex) yield return null;
                    yield return "Viola";
                    count++;
                }
            }
            Assert.That(() =>
             InfiniteSeqRosa().MinUpToNow(InfiniteSeqViola().Take(100)).ToArray(),
             Throws.InstanceOf<ArgumentException>());
        }
    }
```