C#中IEnumerable、ICollection、IList、List之间的区别

首先我看看 IEnumerable：

```
// 摘要:
//    公开枚举器，该枚举器支持在指定类型的集合上进行简单迭代。
//
// 类型参数:
//  T:
//    要枚举的对象的类型。
[TypeDependency("System.SZArrayHelper")]
public interface IEnumerable<out T> : IEnumerable
{
  // 摘要:
  //    返回一个循环访问集合的枚举器。
  //
  // 返回结果:
  //    可用于循环访问集合的 System.Collections.Generic.IEnumerator<T>。
  IEnumerator<T> GetEnumerator();
}
```

IEnumerable<T> 实现IEnumerable接口方法，那IEnumberable做什么的，其实就提高可以循环访问的集合。说白了就是一个迭代。

再来看看ICollection:

```

// 摘要:
//    定义操作泛型集合的方法。
//
// 类型参数:
//  T:
//    集合中元素的类型。
[TypeDependency("System.SZArrayHelper")]
public interface ICollection<T> : IEnumerable<T>, IEnumerable
```

原来ICollection<T> 同时继承IEnumerable<T>和IEnumerable两个接口，按我的理解就是，ICollection继续它们2个接口而且扩展了方法，功能强多了。

我们继续看IList:

```
public interface IList<T> : ICollection<T>, IEnumerable<T>, IEnumerable
```

IList 继承它们三个接口，怪不得功能这么多啊

最后来看看List:

```
public class List<T> : IList<T>, ICollection<T>, IEnumerable<T>, IList, ICollection, IEnumerable
```

它们都是接口，只有List 是类，不仅实现它们的接口，而且还扩展了太多的方法给我利用，几乎所有功能都能实现了。

按照功能排序：List<T> 《IList<T> 《ICollection<T>《IEnumerable<T>

按照性能排序：IEnumerable<T>《ICollection<T>《IList<T>《List<T>