11.30

验证一个系列中一个元素是否存在，应用他们共同存在的性质，与子集（共同存在的）性质

1+2+3+4+5=15

1+2+4+5=12

15-12=3 absent

12.1

Int 与integer的区别

java提供了两种类型：引用类型和原始类型（内置类型）。int是java的原始数据类型，Integer是java为int提供的封装类。  
　　java为每一种数据类型提供了自己的封装类：  
　　原始数据类型 封装类  
　　int Integer  
　　boolean Boolean  
　　char Character  
　　byte Byte  
　　short Short  
　　long Long  
　　float Float  
　　double Double

int与integer的区别从大的方面来说就是基本数据类型与其包装类的区别：

int 是基本类型，直接存数值，而integer是对象，用一个引用指向这个对象

1.Java 中的数据类型分为基本数据类型和复杂数据类型

int 是前者而integer 是后者（也就是一个类）；因此在类进行初始化时int类的变量初始为0.而Integer的变量则初始化为null.

封装类与范型的实现：

Java的泛型实现机制就是这一要求的。泛型参数必须是一个类，即Object类的子类。  
  
但是Java中，int属于原始数据类型，并不是一个类，也就无法使用于Java泛型。  
所以只能用Integer类。

12.12.2017

类的加载与static

Static 代表这个东西无关对象，为“静态的”，即无论你对象怎么变，它都是一样的。

加载一个类时，首先会加载静态变量、静态初始化块，然后会加载变量、初始化块最后再加载构造器

、、、、、

实例化子类的时候，若此类未被加载过，首先加载是父类的类对象，然后加载子类的类对象，接着实例化父类，最后实例化子类，若此类被加载过，不再加载父类和子类的类对象。接下来是加载顺序，当加载类对象时，首先执行静态属性初始化，然后执行静态块，当实例化对象时，首先执行实例块，然后执行构造方法，至于各静态块、实例块之间的执行顺序，是按代码的先后顺序。

<http://blog.csdn.net/happy_horse/article/details/53414565>

求解线性递推关系：

这里的递推关系指的是序列（数列）中的项由其前项来表示的关系，如果该关系室友前项的线性组合来表示的，则称其为线性递推关系，它是k阶的代表它由前k项来表示。

1. 求解常系数k阶齐次递推关系

即找到形如an =rn的解,即它的k阶特征方程的根r1,r2,r3,…,rk满足an =*α*1r1n+*α*2r2n+…,通过初始条件求解m,n,…，即可得到递推关系的解an

*an* = *(**α*1*,*0 + *α*1*,*1*n*+· · ·+*α*1*,m*1−1*nm*1−1*)rn*1

+ *(α*2*,*0 + *α*2*,*1*n*+· · ·+*α*2*,m*2−1*nm*2−1*)rn*2

+· · ·+*(αt,*0 + *αt,*1*n*+· · ·+*αt,mt*−1*nmt*−1*)rn*

1. 求解常系数k阶非齐次递推关系

先求解其相伴的其次递推关系。

然后找到非齐次的一个特解，把他们相加，即为非齐次的解。似乎想到了线性代数里面某个非常相似的东西，不过忘了是什么了，难受。

齐次的递推关系怎么解已经知道了，那么该如何求得这个特解呢?

我们可以通过Fn得到特解的形式。

Suppose that {*an*} satisfies the linear nonhomogeneous recurrence relation

*an* = *c*1*an*−1 + *c*2*an*−2 +· · ·+*ckan*−*k* + *F(n),*

where *c*1*, c*2*, . . . , ck* are real numbers, and

*F(n)* = *(btnt* + *bt*−1*nt*−1 +· · ·+*b*1*n* + *b*0*)sn,*

where *b*0*, b*1*, . . . , bt* and *s* are real numbers.When *s* is not a root of the characteristic equation

of the associated linear homogeneous recurrence relation, there is a particular solution of the

form

*(ptnt* + *pt*−1*nt*−1 +· · ·+*p*1*n* + *p*0*)sn.*

When *s* is a root of this characteristic equation and its multiplicity is *m*, there is a particular

solution of the form

*nm(ptnt* + *pt*−1*nt*−1 +· · ·+*p*1*n* + *p*0*)sn.*

生成函数：

1. 定义
2. 生成函数中可以利用到的有关幂级数的一些公式
3. 计数问题与生成函数
4. 使用生成函数求递推关系
5. 使用生成函数证明恒等式

12.14.2017

Java中new关键字到底有什么用？

之前的理解：为一个object初始化，即在堆空间中开辟其对应的空间，堆空间中的值为null或者0；

问题：我在初始化一个数组a时，a的类型为类B，用语句

B[] a = new B[n];其中B为我自己写的一个类，他拥有的属性为：int data,int position;

这个时候，数组a中的其实并没有B，每个下标存的都是null,也就是说，我调用

a[0].data是会报错的，NullPointerExpectation

但实际的想法是调用a[0].data，返回的值是0

那么要如何实现呢？之前我的做法很简单，写一个遍历a的for循环，每个为它创建一个new B();

但之前在网络上查看有关静态内部类与非静态内部类是看到这样一个操作

OuterClass.InnerClass innerObject = new OuterClass().new InnerClass();

其中InnerClass是out的非静态内部类

new OuterClass().new是什么操作？

先理解一下嵌套类：

嵌套类分两种，一种是静态的一种是非静态的。静态类是会在类对象加载时被加载，无法访问类中的其他成员，而非静态类可以访问类中的任意成员。

什么时候会用到嵌套类：1.当某个类只会在一个类里用到时。2.为了提高封装性。（B在A中调用A的属性，但是A的所有成员依旧可以是private 的）3.让代码可读性，持续性增强。

创建内部静态类的实例：OuterClass.StaticNestedClass nestedObject =

new OuterClass.StaticNestedClass();

创建内部类的实例: OuterClass.InnerClass innerObject = outerObject.new InnerClass();

由此看出：静态类是共有的属性，非静态类想要实例化必须得有外部类的实例。

然后看这么多，应该知道了，上面那个网上的人的帖子是错的new OuterClass().new InnerClass();是不存在的，真尼玛坑爹啊。

老老实实for循环吧。

1月17日

排序：

快速排序属于

4.18

即使不写extends Object 每个类仍然是它的子类，

Object类的方法有：都是非静态的

What is a hash code?

Computation on the value, same reference return the same value;

What is super class of Error and Exception?

RunTimeException;

What is the difference between HashMap and HashTable?

Hashtable is synchronized.(different when you have several threads, whenever you don’t have synchronization, just use unsynchronized)