1. 基础

抽象数据类型ADT API

理解问题 定义问题 控制问题的复杂度 分解成子问题

原始数据类型

语句 声明 赋值条件 循环 调用 返回

数组

静态方法 字符串 标准输入输出

数据抽象封装（面向对象）

二分查找 rank main

标识符 运算符 字面量 表达式（字面量或者变量，或者能运算出结果的式子）

运算符重载 （对浮点和int和char不同）

关于运算符优先级——编译实现

**中缀表达式 是如何求的**

类型转换，自动提升为其他数据类型

比较：产生布尔值

Int 4字节32位 long 8 short byte1 float 4

语句：声明。 赋值 条件 循环 调用 返回

简化：声明并初始化; 隐式赋值 ++ -- +=; 代码段花括号省略（不要省略）

For和while

数组：声明 创建new double[n] 初始化元素 可以简化直接初始化

异常：数组过界访问

复制数组 找最大 有个max 反转 和与 平均值sum+=a[i]

矩阵乘法

二维 int[][] =new int[n][m]

Java中的方法： 基本数据结构 数组 字符串 互相转换（类型转换） 增删查改

绝对值 是否素数 isPrime 必须大于等于2

平方根牛顿：

三角斜边长a2+b2 调和级数

方法的重载

参数是按值传递，传的是地址值或者实际值

Void的副作用 没有返回值但是对外部变量进行了修改。或者调用了系统函数或者其他API来

递归：有一个终止条件（**最简情况，最下层，最终值**），递归调用是为了解决**规模更小的子问题** 递归可以层层返回 或者层层修改，向上传递数值或者向下不断执行。。。 递归存在参数的演变，参数会发生某种变化，缩小，放大，或者是指针嵌套。。需要这种变化规律

递归有单向递归和多向递归，可以有多个出口 ，因为调用后会返回到调用处，继续向下执行 非尾递归是会释放堆栈的。。。尾递归也会但是可以优化成go结构

递归二分

String StringBuilder Integer Arrays.sort（）

Math.random PI E自然对数 abs max(a,b) e double

用泛型或者是重载多个类型参数来实现

2、本书API

随机数： 初始化种子 random0-1 uniform() 0-n-1 start到end

返回bool，概率为p

返回值i ，概率为p=a[i] 分布

正态分布：期望 标准差

数组乱序shuffle

数组最大 最小 平均mean 采样方差var 标准差stddev 中位median

字符串 指向常量池 指向对象 +连接会发生自动类型转换

类型转换toString parseInt()

API接口：分离细节，只管调用就是了

输入输出 标准输入流和标准输出流 java javac more

Print pintln

格式化输出(插值字符串) printf (“”,int,int ) % 2f 输出的字符串长度/占位符类型 长度带小数点表示小数长度，空位0会以空格补

标准输入 读取 一个字符，返回该字符 char a = readChar() 从输入流中读取

读取一行 读取所有

重定向输出流给文件 可以从文件中输入 可以输出给程序（函数）

3、绘图库 点 直线 文本 圆 长方 多边 Line point text circle 画布大小

4、泛型

**避免溢出，进行参数检查 中间值检查**

变量无穷大：DOUBLE.

比较时会发生类型转换x < 3.1

变量没初始化使用会异常

除0异常和 /0.0的区别

包装类型 怎么使用？可以用运算符吗？

商向0取整（偏向于0） 除以负数的余数 把商\*除数，剩下的差

If else二义性，最近匹配

While没有计数器 需要自己定义该变量 当然停止条件不一定是看计数器

6、练习题： +-\*/和比较 < > 的隐式转换 字符连接的转换

命令行参数

f=0 g=1十五次f=f+g g=f-g 1 0 1 1 21 32 53 oldf=f oldg=g f=oldf+oldg g=f-oldg=oldf =>f=oldf+oldg=oldf+oldoldf 斐波

t=(9/t+t)/2

**小于号的情况下会循环几次，计数器最终值**

for（i=0;i<5;i++） 只要满足条件就会执行 计数器增加和后续语句 0 1 2 3 4 执行5次 最后i=5 i<n的话 执行n次 最后=n 最后一次在**循环体中**时值为n-1

**如果**<= 执行n+1次

转二进制 字符串输出 用空字符串加上数字 /2 就是移位 %2就是末位的值，得到二进制各位

同理 /10移位 %10是未位的值，可以求十进制各位

打印二维数组，换行（参数是二维数组），如何获取x,y？

转置数组

A[i]=9-i a[i]=a[a[i]] 交换

Floor(lg2N) 2 ^x =N

计数数组histogram 长度为m,. 0-m b[i] 是i 在a数组中出现次数 b[i]=countKey(array，key)

ExR() return 2n+ exr(n-3)+ exe(n-2) 类似斐波

小心调用堆栈溢出

Mystery b==0则return0 即b<2时 b%2==0 偶数 return mystery(2a,b/2) 否则mystery(2a,b/2)+a 奇数/2 永远是奇数 偶数/2最后会是奇数

**最后一层的返回值是第一层的返回值**

三个数排序 斐波和阶乘的 递归实现，迭代实现 first=1 second =1 for (i<n) result=second+first first =second second=result

二项分布 Binomina(n k p) n k 都==0 返回1 任何小于0 返回0 (1-p)bino(n-1)+p bin(n-1 k-1)

二分查找： 删除重复 count键数量

N\*N布尔数组 ij互质(没有公因子，公因数为1 ) a[i][j]为 true

直方图

分割N边形 圆上平分n个点， 每两点之间概率p连接（可能连可能不连）

矩阵库 向量点乘dot 矩阵积 转置 乘向量 矩阵是二维数组 向量是一维

模拟扔色子， 两个色子之和的古典分配 都是1-6 所有和存在的状况 count[i+j]++

P[i] = count[k]/36

顺序查找（暴力）

1.2 数据抽象 引用类型 class **类型是多组值的集合，比如int 是2^32的集合 float是0.00000000到9.9999999的集合 字符是 ‘a’-z 和符号的集合，实际上都是二进制数**

**可以包含基础值int float等**

Counter（id） id, number increament +1 tally() 增加的次数 toString() 对象转字符串（可重写）

Equals hashCode

对象 状态（属性） 标识（内存位置，区别于其他对象） 行为（操作）、

**数组本身不是对象，没有任何方法 需要用Arrays方法或者转数组列表**

ArrayUtils .reverse .removeElement addAll

数组处理 Arrays 转字符串打印 .toString binarySearch sort 转数组列表asList

填充 fill .equals(a1,a2)

数组列表ArrayList add remove可以是任何类型 addAll(列表) toArray removeAll(列表)求差集 交集retainAll() 查找 contains .size()

集合Collection .max min

转set asList Set<String>

字符串处理 String length() 索引的字符 charAt(i) 字符位置indexOf() 连接 concat 用+代替

分割子串 substring() 分割成数组split 比较compareTo equals

判断是否回文：“ 对称比较

判断是否有序 compareTo 》0

提取文件名 ，提取url的部分

Println(object) 会自动触发 toString

对象变量赋值，只会改变引用 **别名**

对象可做参数，返回值

ParseInt

3、lang包 swt net包 io包 包装类 集合类 符号表ST StingST

Interger Double String StringBuilder 颜色字体

计数器 累加器Accumulatar 计时器StopWatch 日期 Date

队列 包 优先队列 索引优先队列 IndexMinPQ

无向图 有向图 Graph Digraph 边Edge DirectedEdge

MST最小生成树 最短路径 SP CC连通分量 广搜

日期“day month year() toString() amount()金额

4、什么是流 在JAVA中流是一个对象 应该有个buffer 在构造函数 Stream( s )方法中，把数据存入buffer 。 read方法实际上是get方法 write方法则是 修改其他变量值 从buffer里面输出 isEmpty readInt readDouble close

Out print()

**每个类都写个static静态 Main方法用来测试 ，可以无参 或者带命令行String[] args**

对象属性可以用final 构造函数不定义就会默认创造出来

方法中如果 属性和参数名有冲突。可以用this 其他任何时候都不需要this

累加器 Accumulator total N addDataValue() mean() 求所有的平均值 toString()

设计ADT 1、封装数据 2、设计API

3、接口继承 接口就是函数命名 比较接口Comparable 迭代器 Iterator

4 、getClass()返回Class对象 反射 toString hashCode散列值

默认的toString()是对象地址。多重写

封装类型 Byte Character Boolean Long Short

对象==对象 是比较引用

对象比较： Obj1.equals(obj2) null时返回false

孤儿对象：对象在堆上

当变量引用修改， 原对象没有引用，就会被回收 当函数结束。局部变量消失， 这时对象也失去引用

不可变数据 final 对于对象变量，可以改属性，不能改引用

异常，和断言

内存出界 栈溢出 超出数组索引 空指针异常

抛出错误 throw new RuntimeException(“错误信息”)

Assert 条件 : “错误信息” 契约式设计 前条件：何时才能调用。后条件 返回时必须如何 副作用： 方法对对象属性进行的改变 用断言测试

空指针异常，其实是空引用异常 变量为null时，使用方法或者属性

引用是如何实现的：不同jvm不同，用指针（地址） 或者句柄（指针的指针） 方便回收

代码中可以用包名代替import

题目： 正方形生成n个随机点 interval

VisualCounter +1 -1有最大值max

有理数类Rational 和同样对象的计算 plus minus times()积 equals()比较 toString() divides() **断言防溢出**

累加器 方差 标准差 N total

Date 解析字符串类型的日期

1.3 背包 队列 栈 泛型 迭代 链表 使用泛型 继承迭代器接口

泛型在类名后 <T> 然后在方法参数或者返回类型中用

在创建实例时 需要填入真正的类型 这样才能在方法中使用该类型

**泛型时必须用包装类型 但是可以用 + = 逻辑等，会自动拆箱**

下压栈 放邮件 网页回退 界面返回？

算术表达式求值 （1 +2） 双栈求表达 Evaluate

**构造函数没有类型**，因为没有返回值

**基本类型会自动初始化吗？**

**栈要防止溢出**

定容栈（固定容量）需要检测是否满

动态数组栈，可以扩大 栈大小 当栈满了就把栈扩大resize

对象游离：当一个对象的引用不再需要，会继续占用（比如之前的栈数组中的对象，没有删除掉） 可以设为null

集合类型 foreach迭代 冒号 for(int n ：collection )

Interator() { return new 内部类迭代器。 }

相当于while+迭代器 Iterator<> i= collection.iterator(); 迭代器对象有个特殊的泛型数组 ，有内部指针 i.hasNext() 有没有到底 s=i.next(); 下一个数 remove()

Implements Iterator<Item> 迭代器内部数组保存

**Next()指针加一。返回当前值，类似于pop hasNext（ 当前是否到底**

迭代器 方法需要 自己实现， 可以作为内部类存在 import util.Iterator

内部类可以访问外部类的属性 实现Iterator

外部类实现 可迭代Iterable

**泛型在类上定义，在方法和变量中使用**

链表节点也可以用内部类来实现 Node next

链表栈和队列 top指针 first last指针

复合结构：可以用背包实现栈 用队列存储数组 **利用泛型可以把更多类结构传进去**

**实际上泛型里面还可以存数组，栈等复合结构**

数据结构： API+mian用例

用数组或者链表形成结构

父连接树unionfind 二分查找树bst 字符串（数组实现） 二叉堆（对象数组实现）

散列表（拉链） 链表数组 散列表 线性探测

邻接链表Bag对象数组？

单词查找表

不允许泛型数组的原因：共变数组，类型擦除

如何创建栈数组：stack<>[] =(stack<>[]) new stack[] **使用已有的泛型类不用object**

内部类的好处：直接使用外部类的属性 stack$node.class

Java.util.stack接口太宽，可以获取任意数据，可以从栈底增加

**集合类中允许插入null，空对象**

**Foreach可以访问普通数组，但是没有迭代器，iterator 字符串不能foreach**

题目： isFull

用栈判断括号匹配是否完整：遇到右括号弹出 Parentheses []{}<>()

N>0时循环： n%2入栈 n/2 ：读入二进制数末位，移位 转2进制

循环：队列不为空，全部入栈 ; 2栈不为空，全部入队

getTop peek 返回栈顶，不弹出

补全中序表达式左括号

中缀转后缀

打印后缀结果

计算中序

实现栈 copy 克隆功能 （深）

数组队列， 定长数组，动态变化的数组 resize(到头了数组\*2，<1/4 数组/2)

如何入队？全部后移 队列元素数量total决定尾索引 数组容量capacity

输出queue 倒数第k个

Current=current.next 遍历下一个节点 所有节点都是这样取的

X．Next = x.next.next 这是**删除中间节点 ，删除下一个**

删除链表尾结点，尾结点的next == null

删除第k个元素 delete(k) 需要计数器 i

Find(key) 查找链表元素 返回true false

t.next =x .next x.next =t 插在x和x.next中间，顺序先要插进去，不要先断了链条

removeAfter（node） 删除后一个结点 insertAfter(node1,newnode) 新节点插入

remove(key)删除所有key 节点 max()返回最大值（循环和递归参数用next）

循环链表：实现队列，非空时last.next=first、

反转链表，返回头结点 异常：注意链表为空时，链表只有头时。边界，头尾结点

创建一个新头，把原来的尾插上去就行了

双向链表： 双向节点 doublenode prev next 表头插入，表尾插入 表头删除表尾删除

节点前插入，节点后插入(根据key找结点)

插入时注意链条不能断，可以先断一节//每个节点有两个链条 prev next 可以先断prev 先接新的，再断旧的 new.next= current.next

栈队列：push pop enqueue 链表实现

双向队列 pushLeft pushRight popLeft (用双向链表和动态数组实现)

随机背包： RandomBag 随机取出。。add 只能用迭代器取 ，迭代器初始化应该打乱背包数组 shuffle

随机队列 RandomQueue (数组实现，动态大小) 正常顺序放入， 出队时随机交换末位并出队 sample（）取出 randomqueue<card> 发牌

随机迭代器 迭代器就是类似pop但是不出队不改变原数组？ 返回当前值

杀人游戏，不被杀的位置 圈 0 – n-1 从第一个人开始报，报到m杀死 队列删除指定结点 循环链表 删除后连上链条

删除第k个元素 如果等于1 注意下 头指针和尾指针

环形缓冲区：环形队列 空和满 满时等待

前移编码：读字符串，链表保存字符，清除重复（链表方便删除） 最近访问的元素可能被再次访问

用构造函数复制队列 栈

递归打印文件夹下所有文件 子文件夹名 和所有子文件夹下的文件（缩进打印） 使用队列和io.File 所有名字都入队？

文本编辑器缓冲区： Buffer()创建 光标处插入删除(char) delete 光标移动left right size

双栈实现还是队列实现？ 双栈需要把内容倒入到另一个栈 只能push pop

混合入栈出栈指令 入栈 数字 0 123~n-1 出栈- -- ---- 输入排列，判断是否会导致栈溢出。。。。——出栈前必须入栈

队列、栈的连接

双向队列实现双栈

三个栈实现一个队列

快速出错的迭代器 迭代器中不能修改数据集合。抛异常

1.4 算法分析

时间和内存

程序运行多久

问题规模：输入的大小，循环的次数 单次循环的复杂度

Threesum 统计所有和为0 的元组的数量 :三重循环(不重复) a[i]+a[j]+a[k]==0 count++

数组长度 n 需要循环次数 n^3?

如果规定元组不重复，而且没有顺序 j=i+1 k= k+1

计时器 RunTimer() getRunTime()

System.currentTimeMills() now-start/1000.0

用随机数作为输入值

规模（N） 时间T(N) 用XY坐标轴表示 N和T（N） （输入多少参数，循环多少次）

N^3

每条语句花的时间 每条语句执行次数

近似时 忽略较小的项（只执行一次的语句） n取大值

增长级别 常数1（与规模无关，固定） 对数lgN（二分） 线性 N（循环） 线性对数（分治） 平方 立方 指数2^n（穷举查找所有子集，是集合匹配，比如找子串）

很多规模只取决于循环的规模 内循环 成本模型 输入规模

常见函数 向下取整 向上取整floor ceiling 自然对数ln 2为底 调和级数 1 1/2 1/3 1/4 阶乘

近似函数 调和级数的和 lnN

等差数列求和 1 +2 +3 N^2/2

等比求和 1+2+4+8 ~ 2N 后面差距越来越大 大约1.7~1.9？

二项式系数

斯特灵公式

指数函数 (1-1/x)^x 1/e

**加速的方法：排序+有序查找 2-sum 和为0的两个数 归并排+二分找相对值**

Arrays.sort() binarysearch count++ 从头向后逐个查找

3-sum i,j相邻 j=i+1 查找 -（）

暴力算法 和改进算法 Fast

倍率实验

**数组长度可以是变量？** 那么如何在编译时分配？

可以是动态数组吗？ C不可以 java的数组只是一个指针，指向堆 堆上的空间是可以即时分配的。。。C的数组元素都分配在栈上，所以需要确定要开多少栈

但是不可以动态吗？eax = [] Sub esp eax 开动态大小的栈

**输入生成器，进行测试**

**问题规模加倍， 算和上一次实验的比值**

**倍率定率 T(2N)/T（N）**

**大型问题需要算多久**

注意系数 系数可能会很大（执行一次循环花很多时间） 真正的内循环 问题规模比较小的时候可能循环影响不大 多个参数的问题

根据输入的不同，问题的难度可能不同 最坏情况

**使用随机化也可以提高处理速度，用概率来处理问题，最坏情况发生概率小**

均摊分析：操作总成本/操作总数

内存的分析 基础类型的占用 对象实例的内存+对象变量本身（16字节）

对象会被填充成8字节的倍数Integer 8 =4+4

引用变量需要占8字节

对象开销：类的引用 垃圾收集信息 同步信息

对象： 对象开销 name(String引用8字节) count(int)+ 填充

链表对象 对象开销 +额外开销 +data next两个引用 ，data是个Item对象

数组对象 ->对象开销+长度信息+填充——指向的n个int

对象数组 对象开销+长度信息+填充+——指向对象的引用

字符串对象： 字符数组的引用 偏移 长度 散列值

子串：substring 创建新String **但是使用原来的字符数组**

局部变量 栈 new的堆 不是二叉堆？

题目： 证明从N个数取三个整数组合的数 n(n-1)(n-2)/6（归纳）

队列实现栈

两个栈实现栈队列

自动装箱的性能代价

Union-Find算法

动态连通性 渗流到网络通信

输入整数对 每对都相连 pq相连 自反：pp相连

传递pq qr 则pr相连 pq相连则qp相连

保存一定的整数对(a,b)，(c,d))然后输入一对值p,q，通过保存的值证明这两个值是相连的

1、

网络问题：检查p ,q是否相连，外部和什么相连，是否需要建立一条新的连接才能通信

判断变量名等价：指向同一对象的多个引用

数学集合 pq是否属于相同集合

单个数，触点 整数对，连接 等价类-连通分量/分量（几个触点相连组成的集合） 用0-n-1表示 **可以把整数和任何名称关联**

**分量是不和其他分量连接的n个触点的集合**

UF类 UF（n） 初始化触点 从 0-N 添加连接union（p,q） find(p)所在的分量

isConnected(p,q)是否在同一分量中（是否连通）count() 有多少分量

**触点在不同分量时,union会合并分量（开始的时候，每个触点是一个分量，点与点合并，点与线合并。线与线合并）**

**find返回分量的标识符？**

最开始时分量共有n个，每个触点是一个分量 每次合并分量减一

需要保存 已加入的连接 用数组保存——id[i] 下标索引代表每个触点 ，id则是分量的id 最开始有n个分量 n个id ，连接后共用一个id

IsConnected() 检查分量id是否相等，

Union 的实现： find quick-find 所有连通分量的id相同

两个分量p，q连通时，遍历整个数组，查找所有分量id为id[q]的触点，并将id全部改为id[p] 如果已连通则忽略

Count—

**现代计算机每秒执行数亿指令**

Quick-union id[] 是另一个触点的值 根触点指向自己 属于单方向的指针，是一种静态链表

Find: 分量的名字是根触点的名字 while p!=id[p] p=id[p] 不好理解——

换个名字 current = p while(current!=next(current)) current = next[current]

Root = current; return root;

Union： 把根触点连到另一个根触点上

简单理解： if find(p)==p; find() =🡺这个很慢

if p==next[p]&&q==next[q]&&p!=q 连接两孤点 next[p]= q

//包含线相连和点线相连-

rootQ= find(p); rootp= 如果root相等，说明连通 ，跳过 否则

count—

**注意：链表是有向的，p连q和q连是不同的，点连线和线连点也是不同的**

**但是连接本身是无向的，只要根节点相同，就是连通的，可以有多个链条指向同一个root，构成一棵树**

4、表示为森林 可以用节点对象代表 data next

树的大小 是节点数量 节点深度是到root的连接树 高度是最大深度

**大部分树的实现是从根指向各子节点，链接树则是next从子节点指向根，并且可以动态创建根，链接树可以单节点，或者只有一个分支**

5、加权 quick-union算法 每次都把小树连到大树上：

从链表的意义上，这是减少链条的长度，而增加根节点连的链条数量（分支）。。。大树上连的

会越来越多。。。而连到根的距离都会最短，相当多时候是一个单独节点连到根上

权在这里是分量大小

需要再来一个数组记录各分量大小,索引代表分量id（即根id） size[rootid] 初始化为1

比较 p q 的根root size[root] 小的加到大的 连接后尺寸相加 size[rootp]+=size[rootp]

节点深度最多为lgN

8、最优算法 路径压缩的加权quick-union （并不是每一步都最快，但是平均最快）

算成本： 每次连接 用cost记录访问size[] next[]次数，total cost记录总成本 均摊成本=totalcost/操作总数(第n次连接)

题目： 1、删除连接成本比较高 delete()

2cell-probe模型 只记录对随机内存的访问，其他操作忽略 3、输入序列 得到输出