算法(面试):

冒泡: 两两比较，外层循环把最大的移到最后，内层循环依次把更大的往后移一位

选择排序：外层循环按顺序拿个值出来，内层循环将这个值跟剩下的值比较，找到更小值后来到外层循环交换顺序

快速排序：用递归，右边找到比基准数小的数就停下来，否则往左走，在外层循环交换。每次递归最后交换第一个数与基数的位置

归并排序: (二分法) 将数组分割成n个片段，每个片段只含一个元素

二分查找：就是对排好序的数列查找

链表反转：while(q){ r=q->next; q->next=p;//拿到前一次循环的p; p=q;//存p供下次循环使用; q=r }

实现一个字符串的逆序输出.如给定字符串为“hello world”,输出结果应当为“world hello”。(虽说编程语言不限，但是一般还是用c，别用高级语言)

二叉树：先序遍历：根节点->左子树->右子树 中序遍历:左子树->根节点->右子树 后序遍历: 左子树->右子树->根节点

两个数的最大公约数公式: 上次循环除数%上次循环余数

设计模式原则：

1.开闭原则:在不修改原来的代码下扩展 2.里氏设计原则：子类可以实现父类的抽象方法,但不能覆盖父类的非抽象方法，子类可以增加自己特有的方法 3.依赖倒转原则: 4.接口隔离原则 5.合成/聚合复用原则 6.最小知识原则 7.单一职责原则

创建型模式：单例模式, 抽象工厂模式(对工厂的抽象,基工厂有的方法才能在子类中写) ,建造者模式,原型模式.工厂模式

并发设计模式：代理模式,组合模式,桥接模式,享元模式,外观模式,装饰模式,适配器模式

框架级别设计模式: 观察者模式,访问者模式,中介者模式,解释器模式,策略模式.迭代器模式,命令模式,状态模式,备忘录模式,模板方法模式,责任链模式 MVC MVP MVVM

策略模式: if else

桥接模式：将抽象部分与它的实现部分分离，使它们可以独立地变化。

享元模式：用一个享元池节省内存，参考UITableViewCell重用

MVC: Controller import View和Model

MVVM：Controller import View import ViewModel import Model

MVP: Controller import View和Presenter(delegate操作View) import Model

适配器模式: 类适配器：让一个类实现某个协议，让后调用这个类所属对象协议的方法。 对象适配器：类中的某个属性实现了协议的方法 使用条件比如可以传id类型，然后转换(协议)，抽象类遵守协议，子类重写

与桥接模式区别：桥接模式先有桥

消息转发机制：

实例方法都会进入:+(BOOL)resolveInstanceMethod

class\_addMethod(self, sel, IMP(newRun), "v@:") 参数2：方法选择器，参数3:新函数指针；参数4: v void @:object void newRun(id self, SEL sel)

有参数的情况： v@:@ void newRun(id self, SEL sel,NSString \*str)

上个方法没有实现进入-(id)forwardingTargetForSelector

第三个方法，方法签名-(NSMethodSignature \*)methodSignatureForSelector return [NSMethodSignature signatureWithObjcTypes]

进入第四个方法 拿到方法签名: -(void)forwardInvacation

五：doseNotRecognizeSelector

聚合跟依赖区别：聚合部分可以单独用？

构建者模式: 指挥 import 承包商(协议) <- 工人(实现)

中介者模式： 实现抽象Media -> 抽象Media import 多个抽象colleage <- 实现?

迭代器：具体集合类-实现->抽象集合类<-客户端->抽象迭代器<-实现-具体迭代器 具体迭代器-关联-><-依赖-具体集合类

组合模式：比如view和subViews Node subNode

模版模式：就是子类继承父类规定的方法

命令模式：将要改变的对象和变量封装起来。 Invoker-依赖->命令协议<-实现-命令类-依赖->处理器

装饰模式：就是在子类中添加或重写方法

原型模式：clone对象 mutableArray1= mutableArray2对里面的元素是浅拷贝

RAC: pod init pod search 'ReactiveObjc'

RACCommand: execute -> init…Block -> subscribeNext(true) -> executionSignals -> subscribeNext(false)

RxSwift RxCocoa

BehaviorSubject:会接收从前面后面来的订阅，但是只有一个，头一个或尾一个

nameEntryTextField.rx.text.map{…}.debounce(2, scheduler: MainScheduler.instance)延迟2秒响应

asDriver(): 1. 它不会发射出错误(Error)事件 2. 对它的观察订阅是发生在主线程(UI线程)的 3. 自带shared, 一般map后面会加shared防止多次调用

git add remove origin <url> 可能挡在这步git push origin master

github上查看远程仓库版本： release

添加版本git tag –a ‘0.0.1’ –m ‘第一个版本tag’ git tag git push –tags 或提交新版本git push origin 0.0.2

删除tag： git tag –d 0.0.1 git push origin :0.0.1

cocoapod目录： 资源库->Caches->Cocoapods 更新本地索引库pod setup

创建spec: pos spec create LYTest spec文件中s.description的长度要比s..summary长 将远程地址放到s.homepage和s.source s.license=”MIT”

s.source\_files过滤,安装哪些文件 注释s.exclude\_files

pod trunk register orta@cocoapods.org 'Orta Therox' --description='macbook’

pod trunk push LYTest.podspec