分布式算法

第一章导论

Q1:什么是分布式系统?

> 一个分布式系统式一个能**彼此通信**的单个计算装置的集合(计算单元:硬--处理器,软--进程),包括**紧耦合系统(共享内存)**,**松散系统**(cow,Internet(使用互联网作为通信的协议))

总结: 彼此通信的计算装置集合

Q2:分布式系统与并行处理的区别?

> 具有更高程度的**不确定性**和**行为独立性** 并行处理的目的是使用所有处理器来执行**一个大任务** 分布式系统指每个处理器有自己**独立的任务**,但彼此之间需要协调动作(出于共享资源,可用性,容错等原因)

总结: 并行处理——**共同任务**,分布式系统——**独立任务**但需要协调

Q3:分布式系统作用?

> - 1.共享资源
> - 2.改善性能:并行求解问题
> - 3.改善可用性,提高可靠性,防止某些成分发生故障

总结: 方便(共享资源),快(改善性能),可靠(防止宕机导致系统不可用)

Q4:分布式系统当前面临的困难?

> - 异质性:软硬件环境,计算介质不同
> - 异步性:时间发生的绝对甚至相对时间不可能总是精确知道
> - 局部性:每个计算实体只有全局情况一个局部视图(无法获取全局信息)

总结: 设备多样性,无绝对时间,无全局状态视图,结点随机故障

Q5:分布式算法研究方法

> - 1.对各种分布式情况发生的**问题进行抽象**
> - 2.精确的**陈述问题**
> - 3.设计和分析有效算法**解决问题**
> - 4.**证明算法的最优性**

总结: 抽象,陈述,设计,证明

Q6:设计分布式算法需要考虑的因素?

> - 通信: 计算实体间通过**msg传递信息**还是使用**共享变量**(本课程只考虑msg传递消息)
> - 可用信息: 哪些**计时信息**,**行为**是可用的(体现异步性以及局部性)
> - 容错: 哪些错误是可以容忍的

总结: 通信,可用信息,容错

Q7:分布式算法中有哪些衡量算法性能的指标

- 1.时间复杂度:从算法第一个结点启动,到最后一个结点结束所需要的时间
- 2.空间复杂度:额外的辅助空间,共享变量大小等
- 3.信息复杂度:在基于msg传递模型中,为了保证算法正确性,所需要传递的信息总数
- 4.故障/非故障节点个数(本课程不考虑)

总结: 时间/空间/msg

Q8:常用的分布式抽象模型有哪些? 本课程主要考虑三种模型,按照**通信介质(共享存储/msg传递)** 以及 **同步程度(同步/异步)** 进行划分

- 异步共享存储模型:紧耦合,处理器时钟信号不来自同一信号源(通信介质:共享内存,同步程度:异步)
- 异步msg传递模型:用于松散耦合及其,以及广域网(通信介质:msg,同步程度:异步)
- 同步msg传递模型:**理想的msg传递系统**,该系统中,计时信息(如msg延迟上界)已知,系统划**分为轮执行**,**是异步系统的一个特例**.(通信介质:msg,同步程度:同步)

总结: 同步/异步+共享存储/msg,其中同步的共享存储可以直接理解为并行计算因此不考虑

Q9:分布式系统的错误种类?

- **初始死进程:** 局部算法没有执行,直接跳过
- **Crash failure崩溃错误(损毁模型):** 没有警告在某点停止操作
- **拜占庭错误:** 一个出错可能引起任意动作,执行了与局部算法不一致的任意步.该错误发生的进程可能发送的信息包含任意内容(**设计程序时应该尽量避免**)

总结: 跳过(初始死进程),宕机(崩溃损毁),随机(拜占庭)

第二章 消息传递系统中基本算法

前提:本章研究的是无故障的msg传递系统,考虑两种主要的记时模型:同步/异步

Q1:消息传递系统和的形式化模型有哪些基本概念?

- 拓扑结构:无向图,结点代表处理机,边代表**双向**信道
- 算法:由每个处理机上的局部程序构成,**局部程序**包括:
    - 1. 执行局部计算
    - 2. 发送/接受msg
- 状态:由pi的变量,pi的msgs构成,pi的每个状态由2r个msg构成(r为邻居个数)
    - $outbuf_i[l](1 \le l \le r)$:表示pi经第l条关联的信道发送给邻居,但还未传到邻居的msg
    - $inbuf_i[l](1 \le l \le r)$:表示pi的第l条信道上已传递到pi,但未经pi计算处理的msg
- 初始状态:Qj包含一个特殊的初始状态子集,每个 $inbuf_i[l]$**为空**,但 $outbuf_i[l]$**不一定为空**.
- 转化函数:处理器pi的转换函数,实际上是一个局部程序,将输入缓冲区中的信息情况并产生输出
- 配置:配置是分布式系统在某点上整个算法的**全局状态**
- 事件:系统中发生的事情模型化为事件,对于msg传递系统中主要分为两种:
    - omp(i)--计算事件,代表处理器pi的一个计算步骤,其中pi转换函数被用于当前可访问状态
    - del(i,j,m)传递事件,表示msg m从pi传递到pj

- 执行:系统在事件上的行为被模型话为一个执行
- 安全性条件:表示某个性质在**每次执行**中每个**可到达配置**里必须成立(在序列的每个优先前缀里必须成立的条件),从非形式化语言表示:坏事从不发生,**保证运行合法**
- 活跃性条件:表示某个性质在每次执行中的某些可达配置里必须成立,必须成立一定次数的条件,非形式化的语言描述:最终某个好事将会发生,**保证程序可终止**

总结: 拓扑结构用于描述图链接信息,**算法**为局部程序的集合,局部程序将某个节点的**状态**通过**转化函数**转化为下一个状态,这个过程称之为一个**事件**,对于一个事件进行的局部程序我们称之为一个**执行**,对于所有节点在某事件发生后的全局状态称之为一个**配置**,对于每次执行均成立的性质称之为**安全性条件**(通常情况下我们将该性质定义为执行合法的条件),至少存在一个执行使得某性质成立称之为**活跃性条件**(通常情况下我们需要让程序终止满足该条件)

Q2:如何证明满足安全性? 三种方法

- 1.定义:对于所有$r \in I, P(r)$成立且$P_i \rightarrow P_{i+1}$,断言P总是成立,在每个初始配置中断言P总是成立,并且在每一次转移中保持不变,因此在每个可达配置中不变式成立
- 2.定理一:如果P是S的一个不变式,那么对于S的每次执行的每一配置,P均成立
- 3.定理二:设Q是S的不变式,设$Q \rightarrow P$对于每一个$r \in C$, P在每一执行的每一配置中均成立.
- 其实还有一个一级导出的,太麻烦了就自己看吧

总结: 1.证明**所有**初始状态均成立,且在转移中不变 2.证明为不变式 3.证明存在一个不变式可以推到该结论

Q3:如何证明满足活跃性条件?

- 1.证明有限,即算法不会无限执行下去
- 2.构造一个良基集(不存在无穷递减的偏序序列)
- 3.找到一个范函数,将转移系统的每个配置映射到上述良基集中
- 4.证明不会死锁能正常终止

总结: 这个比较迷,不考的概率比较高,有兴趣可以自己了解

Q4:异步系统有哪些特殊限制/不同?

- msg传递事件与处理器两个相邻步骤之间的**时间无固定上界**.例:e-mail传递时可能需要数天,因此设计异步算法时需要**特殊的计时参数**,不能依赖该上界.
- 执行片段:与上述定义不同,这里的执行片段表示的是指一个配置以及事件交替构成的序列
- 执行:这里的执行要求第一个配置必须为初始配置
- 调度:执行去除所有配置之后的序列,如果局部程序是确定的(无随机),则执行可以由初始配置以及一个调度唯一确定
- 容许执行:对于拥有无限计算事件的异步系统中来说,其容许执行表示每个计算的msg都最终被传递(处理器无出错),即消息不会丢失

总结: 无

Q5:同步系统有哪些特殊限制/不同?

- 同步执行分轮进行执行,**每轮由一个传递事件以及一个计算事件组成**(先进行消息的传递,后进行计算任务处理传递的信息)
- 容许的执行:指无限的执行,因为轮的结构因此每个处理器可以**执行无限数目的计算步**,同时每个被发送的**msg最终被传递**

Q6:同步系统与异步系统的区别?

- 在一个无错的同步系统中,**一个算法的执行只取决于初始配置(同步系统)**,但在异步系统中对于相同的初始配置以及无错假定,**由于处理器步骤间隔以及消息延迟的不确定性,因此对于相同算法可能由不同的执行.**

总结: 异步算法需要考虑消息传递延迟导致的先后区别

Q7:什么是算法的msg复杂性?

- 算法在**所有容许执行**上发送的msg总数的最大值(包括同步系统以及异步系统)
- 衡量标准:
  - 1.消息条数(不考虑消息位数)
  - 2.消息总位数(如2个8字节的消息,比一个64位的消息更小)

Q8:什么是算法的事件复杂度?

- **同步系统:** 最大轮数,算法的任何容许执行知道终止的**最大轮数**
- **异步系统:** 首先我们基于下面两个假设
  - 假设1.节点计算任何有限数目事件的时间为0
  - 假设2.消息发送和接受之间的时间假设为1
- 那么基于上述两个假设的情况下,异步系统的时间复杂度为所有计时容许执行中到终止的最大时间.

Q9:异步算法中什么是计时执行(timed execution)

- 每个事件关联一个非负实数,表示事件发生的时间,时间起始于0且非递减(**对于单个处理器时严格增的**)因此执行的时间可以**根据时间进行排序**

总结: 可能存在时间戳相同的事件(因为异步),单处理器上不存在相同事件戳的不同事件

Q10:如何在生成树上进行广播?

- 0.假设生成树给定,且对于生成树上相邻节点之间存在双向信道,每个节点使用变量terminated标记是否终止
- 1.初始状态:在根节点的所有孩子节点的outbuf中存放消息M准备发送
- 2.计算事件:当某节点的inbuf中存在消息M,则将其放到所有孩子节点的outbuf中准备发送,并进入终止状态
- **消息复杂度:** 在同步或异步模型中,msgM在生成树的每一条边上恰好发送一次,因此msg复杂性为O(n-1),对同步/异步模型均适用
- **时间复杂度:** 设h为树的高度,复杂度为O(h),对同步/异步模型均适用

Q11:如何在生成树上进行敛播/汇聚?

- 1.由叶子节点启动,每个处理器p_j计算以自己为根的子树里最大值v_j,然后将v_j发送回p_j的双亲
- 2.对于每个非叶节点j,设p_j有k个孩子,等待k个孩子msg均受到之后,将最大值发送回p_j的双亲节点
- **消息复杂度:** 在同步或异步模型中,msgM在生成树的每一条边上恰好发送一次,因此msg复杂性为O(n-1),对同步/异步模型均适用
- **时间复杂度:** 设h为树的高度,复杂度为O(h),对同步/异步模型均适用

Q12:如何构造一个生成树?

- 1.设p_r为特殊处理器(根节点)
- 2.从p_r开始,发送M至所有邻居
- 3.当p_i第一次收到信息(从p_j处收到),则将M发送至除了p_j以外的所有邻居节点
- **msg复杂度:** 每个信道上消息M至多被发送2次(除了根节点的信道只有一个消息),因此消息复杂度可以达到$O(E)$,在完全图的情况下边数E可以达到 *$\frac{n(n-1)}{2}$* ,因此最坏情况下msg复杂度为$O(n^2)$
- **时间复杂度:** 对于同步/异步算法时间复杂度均可表述为$O(D)$,D表示图的直径(任意两点之间距离最大者),最差的情况下退化为链,此时时间复杂度可以表述为$O(n)$

Q13:如何证明Q12的异步算法构造了一颗生成树?

- 1.证明联通:反证法若存在一对邻居节点,i能从根节点到达,但j不行,证明算法流程中j会设置i为parent,从而联通
- 2.证明无环:反证法若存在环,则设环内第一个收到信息的节点为i,证明存在一个环内节点先于i收到msg,与前提矛盾

Q14:证明Q12的同步算法构造的是一颗BFS树?

- 按轮进行分析,使用归纳法来证明,初始情况满足为BFS树,当k轮生成了一颗BFS树,证明k+1轮也为生成树 PS:由于异步系统存在msg传送延迟,因此可能生成的图不是BFS树

Q15:如何构造指定根的DFS生成树?

- 1. 设定P_r为指定的根结点,P_r从还未向其发送消息的邻接节点中任选一个发送消息<m>.
- 2. 当P_i从P_j收到消息<m>是第一个来自于邻接节点的消息时,P_j为P_i的双亲,向其发送<parent>消息后,对所有之后对其发送消息的其他节点发送<reject>
- 3. 当P_i还未发送消息的邻居中任选一个,发送消息<m>,等待回复<reject>或者<parent>消息,并将发送<parent>的节点加入到自己的孩子中
- 4. 当P_i向所有邻居节点都发送过消息后,P_i终止
- **时间复杂度:** 每条边均需要访问,且串行确认,因此时间复杂度为$O(E)$,最差情况下等价于$O(n^2)$
- **消息复杂度:** 每条边上均可能发送<parent>,<reject>消息各一次,因此总的复杂度为O(4m)
- **优化:** 当一个节点确认parent之后,向其所有邻居广播,让其从对应节点的未发送节点中删除,这样消息复杂度会增加为$O(6m)$但是时间复杂度可以降低到$O(D)$也就是$O(n)$
- **总结:** 根节点自发唤醒,其他节点等待收到msg,收到msg唤醒后对之后发送的任何请求回复reject,并从邻接节点中随机转发msg,同时等待回复,当所有邻居节点均回复之后,往父亲节点发送回复消息并终止

Q16:如何构造不指定根的DFS生成树? 假设: 各节点标识符唯一(不妨假设为不同的自然数) **基本思想:**

- 节点可**自发唤醒**,均尝试构造一颗以自己为根的DFS生成树,当两颗DFS树尝试链接同一节点时,将该节点加入**根id较大**的DFS树
- 对于每个节点设置一个leader变量,初值为0,唤醒自己时设置leader为自己的id
- 当节点唤醒时,将自己id发给所有邻居
- 当P_i收到来自邻居P_j的标识符y,比较y(邻居的标识符)以及$leader_i$(自己的标识符),可以分成下面几种状况:

  1. 如果$y > leader_i$(邻居的leader比自己的大):将$leader_i$设置为y,将P_j设置为P_i的父亲,然后修改所有P_i的DFS子树中所有节点的leader

2. 若$y < leader_i$(自己比邻居的leader大):不操作,因为后续操作某一id比y大的树将会更新邻居信息.
3. 若$y == leader_i$,不操作,但发送<already>,防止其停机阻塞,因为同属一个DFS树中

消息复杂度: 对于一个具有m条边,n个节点的网络,其自发启动的节点共p个,其ID最大者的启动时间为t,则算法的消息复杂度为$O(pn^2)$,可以考虑这么一种特殊情况在完全图的情况下,自发启动的节点按照id升序依次将其与结点id值更新为其自身id,每次有$O(n^2)$个消息被转播,一共p次,因此最差情况下消息复杂度为$O(pn^2)$ **时间复杂度:** 时间复杂度为O(t+m),这里的m感觉不太对劲,个人倾向于n

小结: 分类:

- 单源(centralized alg):仅有一个启动者
- 多源(decentralized alg):任意进程均可称为启动者
- 启动者(initiator):自发地执行局部算法,由一内部事务触发执行
- 非启动者:接受一个msg后触发其执行

复杂性:

- Msg复杂性:msg总数
- Bit复杂性:发送msg中bit的总数目

时间复杂性: 分布式算法的时间复杂性是满足下面两个假定的计算所耗费的最大时间:

- T1:一个进程在零时间内可计算任何有限数目的事件
- T2:一个msg发送和接受事件设定为至多1个时间单位

缺点:针对算法的所有计算,其结果可能是极不可能发生的计算(??)

分布式算法的**one-time复杂性**是满足下述假设的一个计算最大时间,同样需要满足下面两个假设:

- O1:同T1
- O2:发送和接受msg所需要的时间恰好为1个单位时间

缺点:某些计算可能被忽略,其中可能有极其耗时的计算.(??)

第四章 分布式系统中的计算模型

Q1:什么是向量时间戳(Lamport)?

- 使用一个向量来表示时间戳,**向量长度为处理器个数**,对于事件e的向量时戳$e.VT[i]$表示在处理器i上需要有$e.VT[i]$个事件发生后事件e才能发生.
- **向量时戳的比较:** $e1.VT \leq e2.VT$当且仅当对于$e1.VT[i]$每一位均小于$e2.VT[i]$ $e1.VT < e2.VT$当且仅当满足上述条件的同时,要求$e1.VT \neq e2.VT$

Q2:怎么使用向量时间戳来保证因果相关? 前置设定:

- 对于每个节点,有一个局部时戳my_VT
- 对于每个事件,有一个向量时戳$e.VT$
- 对于每个msg,有一个向量时戳$m.VT$

具体操作:

- 执行事件:将自己的时戳赋给该事件(对应位更新)

> - 发送msg:将自己时戳赋给所有发送的msg
> - 接受msg:自己向量每一位更新为自己与msg较大者

证明:

概率算法:

Q1:确定算法的平均时间,与概率算法的期望时间有什么区别?

> - 确定算法的平均执行时间:输入规模一定的所有输入实例是**等概率出现**时，算法的平均执行时间。
> - 概率算法的期望执行时间:**反复解同一个输入**实例所花的平均执行时间。

进一步的我们对于概率算法还定义了下面两种期望:

> - 平均的期望时间：所有输入实例上平均的期望执行时间
> - 最坏的期望时间：最坏的输入实例上的期望执行时间

总结: 确定算法平均执行时间针对**不同输入**的期望,概率算法的期望执行时间针对**同一个输入**的平均执行时间(因为有随机因素导致运行时间有差异)

Q2:概率算法主要有几类,分别用于解决什么问题?

> - Numerical(数字算法):用于获得近似的数学解,通常情况下使用概率论以及数值分析的方法,如计算定积分,求集合的势等
> - Monte Carlo(蒙特卡洛):问题只有1个正确的解，而无近似解的可能时使用MC算法,但需要注意的是MC算法并**不能保证得到解的正确性**
> - Las Vegas: LV算法**绝不返回错误的答案**,但是每次执行均有概率失败,因此最坏的情况下可能需要很长的时间才能找到一个正确解
> - Sherwood:当某些确定算法解决一个特殊问题**平均的时间比最坏时间快得多**时，我们可以使用Sherwood算法来减少，甚至是消除好的和坏的实例之间的差别.用于平均运行时间差距过大的问题,通过映射的方式对原始输入进行一个合理的**随机化**,使得具有良好的平均性能.

Q3:数字算法中计算定积分有几种方法,实现原理是什么?

> - hit or miss 算法:给定一个包含积分区间的规则图形(面积好求),随机一个点坐标,判断点是否在积分区间,重复多次,求出点落入积分区间的概率,进而得到积分区间的面积,特别的,当重复次数大于$\frac{1}{4\epsilon^2\delta}$时可以保证相对误差小于$\epsilon$的概率大于$1-\delta$,即给定相对误差上界$\epsilon$,误差过大的概率不会超过$\delta$
> - crude算法: 在积分区间内随机均匀产生点,求出这些点的算数平均值,并乘区间长度即得到积分区间的积分值,其可以理解为,将积分区间划分为若干个小矩形,积分值即为小矩形面积之和,其表达式可以表示为:
>
> $$\int_a^b f(x)dx = (b-a)\frac{1}{n}\sum_{i-1}^{n} f(x_i), a \le x_i \le b$$
>
> - 梯形算法: 与crude算法类似,不过从计算矩形面积优化为梯形面积

Q4:如何计算集合的势(大小)?

> - 1. 从有限集合种有放回的取出元素(与无限集不放回的取等价)
> - 2. 要求取元素均匀且独立

- 3. 设k为第一次抽到已经抽到元素的次数
- 4. 当n足够大时,k趋近于$\sqrt{\frac{\pi n}{2}}$
- 5. 因此n的估计式为$n = \frac{2k^2}{\pi}$
- **时间复杂度:** 期望的复杂度为 $O(\sqrt{n})$,因为第一次出现重复元素k的期望为$\sqrt{\frac{\pi n}{2}}$
- **空间复杂度:** 类似的我们需要建立一个映射,用于记录前k-1次出现的元素是哪些

Q5:如何计算多重集合的势(元素种类数)?

- 设多重集S,将元素使用函数$h(x)$散列为01串
- 设函数$\pi(x)$为,返回01串x第一次出现1的下标
- 对于多重集合所有元素,生成一个01序列ans,生成方案如下:
- ans第k位为1,当且仅当存在一个元素$a \in S$,使得$\pi(a) = k$
- 对于最后答案我们估计多重集合的元素数n为,不妨设$k = \pi(ans)$,估计的n值趋近于:

$$n = \frac{2^k}{1.54703}$$

但我们使用$2^{k-2} \sim 2^k$作为一个粗略的估计值也是可以的

Q6:sherwood算法的复杂度构成?

- 首先sherwood算法基本上是对某个确定性算法,进行随机化之后获得一个平均性能较好的算法
- 设原先的确定性算法平均执行时间为: $$ \bar{t}*A(n)*=*sum*{x\in X_n}t_A(x)/|X_n| $$
- 则由于sherwood算法额外引入的随机化步骤带来的开销,则总体的时间$t_B(x)$可以改写为:

$$t_B(x) = \bar{t}_A(n) + s(n)$$

其中s(n)为随机化取得均匀性所付出的代价,通常情况下远小于确定性算法计算时间

Q7:如何将一个确定性算法改成sherwood算法? 将一个算法改成sherwood算法的一般算法为:

- ① 将被解的实例变换到一个随机实例。// 预处理
- ② 用确定算法解此随机实例，得到一个解。
- ③ 将此解变换为对原实例的解。 // 后处理

总结:步骤一相当于编码器,将输入实例随机化为一个新实例,步骤三相当于解码器,**要求随机化后实例的结果可以转化为原实例的解**

Q8:Las Vegas算法的特点?

- 可能不时地要冒着找不到解的风险，算法要么返回正确的解， **要么随机决策导致一个僵局。**
- 若算法陷入僵局，则使用同一实例运行同一算法，有独立的机会求出解。
- 成功的概率随着执行时间的增加而增加
- Las Vegas算法一般能获得更有效率的算法,甚至有时对于每个实例皆如此,但是Las Vegas的时间上界可能不存在

总结: 每次执行算法独立且均使用随机决策,因此每次执行均有可能成功/失败,最差的情况下一致决策失败,因此可能会导致一直找不到解

Q9:关于Las Vegas算法如何描述算法期望时间? 首先需要定义一些数学符号:

数学符号	含义
$LV(x,y,success)	x是输入的实例，y是返回 的参数，success是布尔值，true表示成功，false 表示失败
$p(x)$	对于实例x，算法成功的概率
$s(x)$	算法成功时的期望时间
$e(x)$	算法失败时的期望时间

> 对于Las Vegas 算法定义其期望时间为:
>
> $$t(x)=s(x)+\frac{1-p(x)}{p(x)}e(x)$$
>
> 上式可以通过这样的方式推导 $$ t(x) = s(x)*p(x)+[1-p(x)]*[t(x)+e(x)] $$ 即:期望时间成功=成功概率*成功时间+失败概率*(失败时间+期望成功时间)

Q10:如何计算模p平方根? 定义:对于x若存在某个y使得$x\equiv y^2(mod,,p)$,则称x为模p意义下的一个二次剩余,y为一个平方根,若找到一个y,我们可以构造另一个数$(p-y)$使得$x\equiv(p-y)^2(mod,,p)$ 也成立(打开二次项易得) 定理:对于$\forall x\in[1,n-1],p$ 为任意奇素数,则有$x^{(p-1)/2}\equiv\pm1(mod,,p)$,由费马小定理可得(费马小定理:$x^{p-1}\equiv1(mod,,p)$),并且 **x为模p的二次剩余当且仅当**$x^{(p-1)/2}\equiv1$ (用于**判定**x是否为二次剩余) 假设已知x为模p意义下的二次剩余,求其平方根:

> - 1. 当p%4=3时,平方根为$\pm x^{(p+1)/4}$,可由上述定理推导
> - 2. 当p%4=1时,平方根可以由下面LasVegas算法得到
>
> > - 1.设$\sqrt{x}$为较小的平方根,设$y_1=a+\sqrt{m},y_2=a-\sqrt{m},a\in N^+$
> > - 2. 随机选取2个a,计算对应$y_1^{(p-1)/2},y_2^{(p-1)/2}$
> > - 3. 如果$y_1^{(p-1)/2}+y_2^{(p-1)/2}\equiv0(mod,,p)$ 则进入下一步,否则回到步骤1(由于定理,因此每一项形如$y_2^{(p-1)/2}$的取值仅有-1,1两种情况,因此有50%的概率使得步骤3成立)
> > - 4. 此时不妨设$y_1^{(p-1)/2}\equiv1(mod,,p),y_2^{(p-1)/2}\equiv-1(mod,,p)$ 则我们有:
> >
> > $$y_1^{(p-1)/2}+y_2^{(p-1)/2}\equiv0(mod,,p)\ y_1^{(p-1)/2}-y_2^{(p-1)/2}\equiv2(mod,,p)$$
>
> > - 4. 不妨设$y_1^{(p-1)/2}=c+d_1\sqrt{x},y_2^{(p-1)/2}=c-d_2\sqrt{x}$ 那么我们可以得到下面的关系:
> >
> > $$2c\equiv0(mod,,p)\ d\sqrt{x}\equiv1(mod,,p)$$
>
> > - 此时d已知,我们只需要找到满足$d\sqrt{x}\equiv1(mod,,p)$ 的一个数啊,使得$da\equiv1(mod,,p)$,则该数a即为方程的一个解,这个数可以枚举验证,也可使用拓展欧几里得求解

Q11:如何进行整数的因数分解(找到一个因数)? 原理:找两个与n互素的整数a和b，使$a^2\equiv b^2(mod,,p)$,且$a\neq\pm b$,则两式相减有$(a-b)*(a+b)\equiv0(mod,,n)$ 那么我们就可以找到$gcd((a+b),n)$,这里不使用$a-b$是因为(a-b)可能为1,即为n的一个非平凡因数 定义(k-平滑):若一个整数x的所有素因子均在前k个素数之中，则x称为k-平滑的。举个例子若一个数可以表示为$3^2 5^1 7^1$那么他是4-平滑的

> - 选定k(k越大后面数字越好找,但是因数分解代价会变大,否则反之)
> - 随机选取k+1个x_i,使得满足$y_i=x_i^2 mod,,p$,且y_i是k-平滑的
> - 对所有y进行因数分解,并记录对应因数次数

- 在k+1组y进行组合找到其中一个组合使得$y_iy_j\dots y_k=p_1^{2m_1}p^{2m_2}\dots p_k^{2m_3}$ 此时我们令$b=\sqrt{p_1^{2m_1}p^{2m_2}\dots p_k^{2m_3}}=p_1^{m_1}p^{m_2}\dots p_k^{m_3}$
- 我们令a为对应的x项乘积即:$a=x_ix_j\dots x_k$ 我们可以简单验证$a^2\equiv b^2mod,,p$
- 此时我们找到对应的a,b即可进行因数推断

Q12:对于一个无偏的MC算法,如何确定给定精度所需的重复次数? 符号解释:

- ε :为算法的优势(即$\varepsilon=p-0.5$)
- δ :为给定的容许错误的概率(即算法以$1-\delta$的概率成功)

定理:设2个正实数之和$\varepsilon+\delta<0.5$ ， MC(x)是一个一致的、 $(0.5+\varepsilon)-correct$ 的蒙特卡洛算法,设:

$$C_\varepsilon=\frac{-2}{lg(1-4\varepsilon^2)}$$

如果调用原先算法至少k次:

$$k=\lceil C_\varepsilon lg(1/\delta)\rceil\ =\quad\lceil\frac{-2lg(1/\delta)}{lg(1-4\varepsilon^2)}\rceil$$

Q12:对于一个有偏的MC算法,如何确定给定容许出错概率,求所需的最少重复次数? 符号解释:

- ε :为算法的优势(即$\varepsilon=p-0.5$)
- δ :为给定的容许错误的概率(即算法以$1-\delta$的概率成功)

$$(1-p)^n\le\delta\ n\log(1-p)\le\quad\log(\delta)$$

由于$\log(1-p)<0$因此我们可以得到:

$$n\ge\frac{\log(\delta)}{\log(1-p)}$$

Q13:怎么使用MC算法解决主元素问题?

- 1.随机挑选一个数组中的元素
- 2.遍历数组得到该元素出现的次数
- 3.如果出现次数大于n/2则找到主元素
- **分析:** 这是一个偏真的算法,出错的概率小于$\frac{1}{2}$

Q13:如何判定一个数为素数? 前置知识(费马小定理): 若n为素数,则$\forall a\in[1,n-1]$ 有$a^{n-1},mod,,n=1$,对其取逆否得到下面一个等价的判定命题: 若n和a为整数,若$\backslash exista\in[1,n-1]$ 使得$a^{n-1}mod,,n\neq1$ 则n不是素数. 因此我们可以得到一个简易的质数检测算法:

- 1. 随机选取$a\in[2,n-1]$
- 2. 检测$a^{n-1}mod,,n\neq1$ 是否成立,不成立则返回不是质数
- 3. 重复上述步骤若干次

但是存在部分数字,其满足条件2的数比较多,因此该算法可能对于这些数字出错的概率比较高,因此存在下面优化算法:

定理:定义数集$B(n)$其中元素a满足下面两个条件之一,其中t表示n-1去除2的因数后留下的奇数,s表示n-1的2的因子数,即$n-1=2^st$

- 1. $a^t,mod,,n=1$,这一步使用快速幂需要进行$log(n)$次乘法,如果乘法使用快速乘(取模)的话,每次乘法需要$log(n)$次加法,因此这一步的时间复杂度为$O(log^2(n))$
- 2. $\exist i\in[0,s-1]$ 使得$a^{2it},mod,,n=n-1$,这一步s最大为log(n),每次操作为也可以使用快速乘,因此时间复杂度同上

若n为素数,其对所有$a\in[2,n-2]$ 均有$a\in B(n)$ 类似的,我们可以根据上述结论推导出其逆否命题,若找到一个a不满足B(n)条件,则n不是质数,通常情况下我们重复选取log(n)个a进行测试,因此总体算法的复杂度为:$O(log^3(n))$,同时有定理指出,对于一次测试其失败的概率小于$\frac{1}{4}$,因此这是一个至少0.75正确的偏假的算法

近似算法

Q1:什么是P/NP/NPC/NPH问题?

- **P类问题:** 一类问题的集合，对其中的任一问题，都存在一个确定型图灵机M和一个多项式p，对于该问题的任何（编码）长度为n的实例，M都能在p(n)步内，给出对该实例的回答（yes）。即确定性算法在多项式时间内可解的问题.
- **NP类问题:** 一类问题的集合，对其中的任一问题，都存在一个非确定型图灵机M和一个多项式p，对于该问题的任何（编码）长度为n的实例，M都能在p(n)步内，给出对该实例的回答（yes）。即非确定性算法在多项式时间内可解的(判定)问题.非确定性算法可以理解为当算法遇到分叉时,可以并行的求解找到产生答案的那个分支 另外的NP问题的另一个定义为:NP问题可以表示为在多项式时间内可以验证的问题(即可以检验解的正确性),即指数时间内求解,或多项式时间内验证.
- **NPC问题:** 对于一个(判定性)问题q，若满足下面性质,则称该问题为NPC问题
  - (1) q∈NP
  - (2) 所有NP问题均可多项式时间多一归约到q 则称问题q为NP-完全的(NP-complete，NPC)
- **NPH问题:** 与NPC问题类似,但是不满足条件1

Q2:什么是规约?

- 假设L1和L2是两个判定问题，$f(\cdot)$将L1的每个实例I变换成L2的实例$f(I)$。若对L1的每个实例I，I的答案为"yes"当且仅当f(I)是L2的答案为"yes"的实例，即:

  $$\forall I，I是L1输出“yes“的实例\Leftrightarrow f(I)是L2输出”yes“的实例$$

  则称f是从L1到L2的多一归约，记作： $L1\leq_m L2$ （传递关系） **直观理解:该传递关系可以理解为求解L1问题不会难于L2,即L2至少与L1一样难** 通常情况下,转换函数$f(\cdot)$的复杂度不在额外说明的情况下,仅考虑多项式时间内可以完成,此时将该规约称之为多项式时间规约

Q3:有哪些常用的规约策略?

- **等价规约:** 证明两问题可以互相规约,即对于问题X,Y

  $$X\leq_m Y,Y\leq_m X 则X\equiv_m Y$$

  经典问题:最大独立集问题 $\equiv_p$ 最小顶点覆盖
- **从特殊到一般:** 如证明顶点覆盖可以规约到集合覆盖,即:

  $$Vertex—Cover\leq_m Set-Cover$$

> - **通过编码:** 通过编码映射的方式来将问题转化为新的问题,如3-SAT问题可以通过编码将其映射到一个独立集问题

Q4:什么是近似算法?

> - 由于NP理论,在有限时间内找到最优解这个问题是不现实的,因此我们希望放松对解的要求,通过允许一定误差的情况下,提高算法的有效性,因此近似算法不再要求总是找到最优解。

Q5:如何定义一个优化问题? 对于优化问题$\prod$由三部分构成:

> - 实例集D：输入实例的集合
> - 解集S(I)：输入实例$I \in D$的所有可行解的集合
> - 解的值函数f：给每个解赋一个值，f：$S(I) \to R$

对于最大值优化问题的最优解σ_{opt}需要满足下面条件: 对于给定$I \in D$ 使得:

$$\forall \sigma \in S(I), f(\sigma_{opt}) \geq f(\sigma)$$

此时我们将$f(\sigma_{opt})$记为最优解的值即$OPT(I) = f(\sigma_{opt})$,有时将σ_{opt}称为最优解

总结: 以最大化问题为例,OPT即解集中最大的那个

Q5:近似算法的性能度量? 绝对性能度量:

> 对于一个优化问题,能找到某个常数k,使得其近似算法A得到的结果$A(I)$,与实际最优解的值满足下面关系:
>
> $$\forall I \in D, |A(I) - OPT(I)| \leq k$$
>
> 但是实际上对于大多数NP-Hard问题,**只有当$P = NP$才能找到绝对近似算法**

相对性能保证:

> 对于优化问题,优化问题Π近似算法A在输入实例I上的性能比$R_A(I)$定义为:
>
> $$R_A(I) = \begin{cases} \frac{A(I)}{OPT(I)} & if,,\Pi是最小化问题 \\ \frac{OPT(I)}{A(I)} & if,,\Pi是最大化问题 \end{cases}$$
>
> 此时定义的$R_A(I) \geq 1$,并且其越接近1越好,若存在ϵ,使得$R_A(I) \leq 1 + \epsilon$那么我们称其为$(1+\epsilon)-$近似算法 PS:性能比一定是大于1的

绝对性能比R_A:

> 若对于所有输入实例,存在一个最小的上界,使得其近似比均小于该值,则称其算法A的绝对性能比 对于多机调度中在线List调度算法其绝对性能比为$2 - \frac{1}{m}$ 而对于离线算法LPT,其绝对性能度量为$\frac{4}{3} - \frac{1}{3m}$

渐进性能比R_A^{∞}:

> 对于输入实例很小时,可能会出现近似算法与OPT相差不大,但是绝对性能比相差很大,因此我们通常忽略这些输入实例较小的情况,特别的如果**问题具有scaling性质的话,绝对性能比与渐进性能比相同**

Q6:如何证明背包问题(或类似的具有scaling性质的问题)无绝对近似性能比?

> - 假设存在绝对近似算法$A(\cdot)$那么对于任意实例我们有:
>
> $$|A(I) - OPT(I)| \leq k$$

> 那么我们将其进行k+1倍的scaling操作得到一个新实例使得:
>
> $$|(k+1)A(I)-(k+1)OPT(I)|\leq k$$
>
> 那么我们经过简单转化可以得到:
>
> $$|A(I)-OPT(I)|\leq \frac{k}{k+1}$$
>
> 若解集为整数,则相当于找到了一个求解OPT的多项式时间算法,即证明了P=NP