1.5万字（含绪论、实验材料和方法、实验结果、讨论、小结与展望）

现在还差：

C和java的历史介绍

结果跑出来

小结写一下

展望and摘要

C语言没有运行时检查机制

摘要

>500字；四部分内容：简要背景；论文要解决的问题(简要说明)；所使用的方法；主要结果；结论与意义。

特别注意：摘要中不要大篇幅地写背景、重要性等内容，这部分不要超过四分之一。学习/了解/介绍已有基本概念、基本原理的内容，不要写入摘要中（不是自己解决问题的方法）。

英文摘要中“本文”不能用This paper，应用This thesis 或This dissertation。

Mwt：思路上要强调，生命起源是一个涉及漫长进化的过程。其中包括化学进化和早期的达尔文进化。实验方法，这些进化的研究方面十分困难。计算机模拟能够弥补这种不足。在计算机模拟研究中引入复杂模型十分必要，可以更具体地反映演化规律，更好地指导相应的实验。然后，谈c的运行效率优势和开发效率劣势，java的开发效率优势和面向对象模拟的自然性

写文章最重要的是题目和摘要（总介绍 为什么要做 做了什么 做的意义），然后是前言详细展开为什么要做这个（外审和答辩是外行）生命起源and模拟50%重要，要让外行都懂，要么逻辑很清楚顺畅，要么要有引用，文献是辅助说明的。然后是结果，意外在讨论（总结结果）中反映，有什么可以继续做的。

Zyj：采用Java模型的扩展优势和模拟自然性优势，在运行效率上无优势，这个需要淡化掉；扩展优势在进阶模型中体现

1 绪论

1.1 引言

地球的生命起源问题，即“地球上的生命是如何以及在哪里出现的？”这一问题，一直是科学家致力于研究的事件之一。学界普遍认定的原始生命完整的形态，是当生命体各部分包括且不限于遗传物质、物理边界、代谢物质组合起来相互作用，形成了一个完整的具有达尔文进化潜力的系统的形态，而细胞是公认的生命的最基本单元。达尔文提出的前生命汤理论，被批判缺乏地质证据且这样的系统中没有明显的自由能来源；20世纪70年代起热液喷口的发现，引出了热液起源生命理论，尤其是本世纪初发现的“碱性喷口”因相对温和的温度和压力等良好条件，被视为一种可能的生命最初形成的场所；外星起源生命理论认为构成生命体的有机分子是由地外陨石带来的，然而这种理论被批判在“转移话题”，即其并没有根本性地解释非生物分子是如何转化成生物分子的；“RNA世界”假说指出，生命起源于同时承担复制和遗传两大重要功能的RNA分子，RNA分子催化自我复制，通过自我剪接产生具有功能活性的RNA分子，通过内含子形成的转座元件不断重排以适应不同环境，随着RNA适体分子的出现，RNA分子结合并排列活性氨基酸分子，形成了早期的蛋白质。最终，RNA分子通过逆转录形成DNA分子，并逐步被更稳定的双链DNA分子取代。上述理论都对生命起源过程提出了一种可能性，然而想要进一步证明各假说的合理性，还需要更多的实验或者化石证据。

由于人们对前生命化学的了解和对生命本质的基本认识不足，且在实验室中构建一个完整的原始细胞是极其困难的，所以生命起源仍然是一个缺乏化石证据的高度推测的领域。并且生命起源是一个漫长的进化过程，不仅涉及化学进化，还涉及达尔文进化，起源过程中任意一个不起眼的时间段在历史上真实消耗的时间都是不可计量的，人们只能通过极少量的地质化石来对这个时间段发生的事件进行推理，这使得研究者通过化学方法在实验室模拟起源过程极度困难。随着计算机技术的发展，通过计算机构建模拟模型来推演进化方面的问题成为一种有力的方法，可以为化学实验提供指导性的帮助。

现有使用的模拟RNA世界的生命起源模型是用C语言编写而成的，该模型虽然在运行效率上表现出色，但因为C语言是一种偏向底层的面向过程的计算机编程语言，在编写模型时需要花费大量的时间精力在防止数组越界等编程方面的问题，而非实现模拟的过程，导致该模型在向外扩展为更复杂的模型时极为困难，也使得生命起源的模拟只能局限在较为简单的模拟环境条件下进行。相较而言，Java语言是一种面向对象的高级编程语言，其集成开发环境如IntelliJ等在智能代码助手、代码自动提示等方面表现十分优秀，在编写模型过程中很容易发现数组越界等编程方面的问题，使得编写者可以更专注于模拟本身。Java语言编写的模拟模型虽然在运行效率上不及C语言编写的模型，但其开发效率和易扩展性却远胜于C语言，研发Java语言开发的生命起源模拟模型将有益于更复杂的生命起源模拟，从而为该领域做出新的贡献。

1.2 RNA世界假说及其研究进展

通过复制和突变进行的自然选择，使生化系统由简单进化到复杂。当代生物学中的复制系统由DNA、RNA和蛋白质组成，其中DNA保存着遗传信息，RNA将遗传信息翻译成蛋白质，蛋白质发挥生物功能，并能为DNA和RNA的工作提供催化酶。然而，这一套复杂的协作系统很难由远古地球上的非生物物质直接形成。因此，40多年前Woese和Crick提出了只需要一种生物分子的“RNA世界”的构想，即RNA既充当遗传信息，又充当支持自身复制的催化剂。此后，它一直是最著名的关于复制系统起源的假说。

Altman和Cech因发现核酶这种催化RNA分子而获得1989年诺贝尔化学奖，这是支持RNA世界假说的重要证据。随后Nissen等人发现制造蛋白质的核糖体也由RNA构成，即核糖体实际上也是一种核酶。此后，科学家们在现存生物体中发现了越来越多的催化RNA分子。另外，在核苷酸代谢中，脱氧核糖是由核糖还原而来，而非从含脱氧核糖的前体从头合成的，也支持RNA先于DNA出现的假说。

但关于这一假说仍然存在的核心问题是，第一个决定性的RNA分子是什么样的？

学界普遍认为能自我复制的RNA分子一定是RNA世界的先驱，因此RNA复制酶是最有可能的候选者，但RNA复制酶必须既是用于复制的模板（因此需要具有天然开放的结构），又是具有活性的聚合酶（因此需要具有稳定的折叠结构），为解决这一矛盾问题，Icarus假设RNA产生的环境经历某种形式的规则振荡行为，例如潮汐循环导致温度循环或沿海地区的潮汐活动。

但分子伴侣的发现又为科学家们提供了新的思路，尤其是在发现了在分子伴侣的帮助下，RNA分子可以在没有RNA复制酶的帮助下进行复制后，Ma等人的研究证明核苷酸合成核酶（以下简称NR）也可能在RNA世界中首先出现。

1.3 计算机模拟生命起源的介绍

1.3.1 计算机模拟生命起源的方法

在生命起源的计算机模拟方面，很多科学家致力于用计算机模拟的分子建模技术来理解第一个生物分子形成的可能化学途径，如量子化学模拟方法通过尝试求解量子力学方程（基于薛定谔方程）来模拟电子与原子系统的原子核之间的相互作用，使我们能够在诸如成键和断键等过程中对电动力学进行建模；分子动力学以经典力学为基础，将每个原子或原子群被视为一个单一的实体，而它们之间的力用势能梯度来表示，既需要输入初始原子位置，也需要输入力场参数，这些参数可以从实验中或者量子化学计算中得到。分子动力学模拟非常适合于模拟由键和非键相互作用控制但没有任何化学转化的系统。而本文中采用的演化动力学模拟方法，则更关注于备选的生物分子在给定环境中是否能够形成进化优势，并由此探索环境因子的影响作用。

1.3.2 计算机模拟生命起源的应用

1.3.2.1 C语言与Java语言的发展史

上世纪六十年代诞生了第一个高级编程语言FORTRAN之后，陆续发明出的语言都围绕着单个体系结构，如UNIVAC计算机、IBM700系列计算机等开发，使不同系统用户间的交流变得困难。因此设计一个通用但与计算机无关的算法语言的需求日渐强烈。经过十余年的更迭，基于ALGOL 60开发的CPL（Combined Programming Langurage）语言被简化为 BCPL （the Basic Combined Programming Language）语言，并由贝尔实验室的Kenneth Lane Thompson，以此为基础设计出了简单而且很接近硬件的B语言，1972年，同个实验室的D.M.Ritchie 在B语言的基础上最终设计出了一种新的语言，即C语言。

C语言充分结合了汇编语言和高级语言的优点，高效而灵活，又容易移植。既精练、接近硬件，又克服了B语言过于简单、数据无类型等的缺点，是一门面向过程的、抽象化的通用程序设计语言，能以简易的方式编译、处理低级存储器，广泛应用于底层开发。

20世纪90年代，硬件领域出现了单片式计算机系统，价格低廉并可以大幅度提升消费类电子产品（如电视机顶盒、面包烤箱、移动电话等）的智能化程度。美国Sun公司的Green项目小组借此机会专攻计算机在家电产品上的嵌入式应用。此时互联网也日渐发展壮大，当时互联网的关键是把浏览器超文本页面转换到屏幕上，并有着向同样的高度互动的远景演变的趋势。业界为了使死板、单调的静态网页能够灵活起来，急需一种软件技术来开发一种可以通过网络传播并且能够跨平台运行的程序。Java的前身Oak语言因其是按照嵌入式系统硬件平台体系结构进行编写的，所以非常小，特别适用于网络上的传输系统，且Oak是一种精简的语言，程序非常小，适合在网络上传输。因此迅速占领市场，被各大公司收购，并蓬勃发展到今天。

1.3.2.2 C语言的计算机模拟在生物领域中的应用

随着计算机算力能力的提升以及生物学研究的发展，九十年代，诸多基于C语言开发的服务生物学领域的软件程序诞生。

如GROMACS（Groningen Machine for Chemical Simulations）于1991年诞生于乌普萨拉大学。其基于C语言开发，集成了高性能分子动力学模拟和结果分析功能，可以模拟具有数百至数百万个粒子的系统的牛顿运动方程。其研发的目的是模拟具有许多复杂键合相互作用的生化分子，被广泛应用于研究蛋白质、脂质和核酸的结构、动力学和热力学性质。

1995年发布的LAMMPS（Large-scale Atomic/Molecular Massively Parallel Simulator）是一个用于原子、分子和粒子模拟的软件包。它可以模拟多种物质类型，包括固体、液体、气体和生物分子等，也可以采用不同的力场和边界条件来模拟全原子、聚合物、生物、金属、粒状和粗粒化体系。通常意义上来讲，LAMMPS是根据不同的边界条件和初始条件对通过短程和长程力相互作用的分子，原子和宏观粒子集合对它们的牛顿运动方程进行积分。高效率计算的LAMMPS通过采用相邻清单来跟踪他们邻近的粒子。LAMMPS被设计为高度可扩展的并行代码，可在单个处理器上运行，也可在成千上万的处理器上并行运行。对于百万级颗粒进行计算，LAMMPS就会变得更加吃力。

2007年推出的开源的通用的粒子模拟软件hoomd-blue针对该问题进行了改进，它是一种用于大规模原子、分子和颗粒动力学模拟的软件。它被设计为高度优化的并行代码，可以利用现代GPU和多核CPU的并行计算能力，广泛用于研究各种材料、流体和生物体系的动力学行为。经过几次大的更迭，确立了以python为骨架，C/C++为扩展的程序结构，支持分子动力学和蒙特卡洛的大规模并行计算。python作为最顶层的脚本和语言，使用OOP的方式隐藏了所用的信息细节，因此抛弃了传统的XML格式文件，极大地提高了读写效率。此外，复杂的体系信息都储存在python的数据结构中，通过抽象结构调用，可以和包括numpy和ctype等数据结构无缝对接，极大地减少了重新处理文本文件的麻烦。

1.3.2 java语言做模拟模型的介绍

如今的模拟模型发展趋势，都是在基本功能完备后, 尽量减少学习难度，使研究人员的精力能集中在科学问题上，而不是花大量时间去编程和研究软件本身。在此思路的指导下，Java语言开发的模拟模型软件也逐渐开始占据市场。

1999年由Matthew Pocock、Thomas Down等人研发的BioJava，使开发人员能够利用Java语言的优势来处理生物学数据和进行计算生物学研究，并逐渐发展成为一个功能强大、广泛应用的开源项目，为生物信息学领域提供了丰富的工具和资源。它提供了一系列功能强大的工具和库，用于处理生物数据、分析生物序列和结构、进行生物信息学建模和仿真等任务。可以读取、写入和处理各种生物数据格式的工具和库，内置分析生物序列和结构的工具和算法，包括序列比对、序列搜索、序列特征预测、蛋白质结构分析等。还提供了用于建立和模拟生物学过程的工具和库，包括蛋白质折叠模拟、分子动力学模拟、代谢网络建模等。这些工具和库可以帮助研究人员理解生物系统的结构和功能，以及探索生物学过程的动力学行为。更可以利用已有的工具功能来处理和分析生物数据。其设计注重可扩展性和灵活性，用户可以根据自己的需求扩展和定制工具包的功能。

2001年推出的MASON是一个用Java编写的多智能体模拟工具包，可以直接作为类库集成到程序中。它提供了用于构建模拟模型的工具和库，包括代理的定义、环境的建模、行为的规划等，被广泛用于研究生态学领域。由互相分离的模型层和可视化层组成，模型层可以通过建立检查点把当前仿真模型存储到磁盘上，并且可保存和恢复上一次仿真结束时的状态。可视化层为仿真提供2D或者3D的可视化工具，还包括一个高质量的随机数发生器。

2004年推出的NetLogo，是一个用Java编写的多智能体建模和仿真平台，旨在帮助用户探索和理解复杂系统的行为。它提供了一个图形化的编程界面，用户可以通过简单的图形化编程语言创建和运行模拟模型，且只需要通过浏览器即可访问软件，经过多年的社区发展，各种学科的模型样例都很齐全，使工作者可以“站在前人的肩膀上”。

2007年推出的Repast Symphony是一个用Java编写的多智能体建模和仿真平台，利用多核处理器和计算集群的并行计算能力，加速模拟的运行速度，使其能够处理大规模、复杂的系统模型。丰富的插件和扩展机制，允许用户根据自己的需求定制和扩展平台功能。用户可以定义代理的属性、行为和互动规则，以模拟真实世界中的个体行为和互动，其直观的图形化界面，使用户可以方便地创建、编辑和运行模拟模型，还支持模拟的实时可视化，帮助用户观察模型的运行和结果。

用户可以通过编写插件来添加新的特性或集成外部工具。

本实验采用在IntelliJ中创建项目从头编写并在本地运行代码的方法，以同时利用IntelliJ优秀的纠错能力及扩展便捷程度，及本地运行最快的运行效率优势。

2 实验材料和方法

2.1模型模拟的对象

根据研究表明，单核苷酸发生聚合形成RNA链的反应可能是在矿物表面进行的，因此模型采用NxN的二维方格系统代替三维的矿物表面，并采用环形拓扑以避免边缘效应。系统中有固定数量的核苷酸前体原料，每个方格代表一个局部小空间，原料、核苷酸和RNA分子散布在该空间内，且分子间的距离都是相同的，可以通过随机碰撞发生反应，即任意两个分子都具有相同的碰撞机会。

核苷酸前体原料可能发生的行为有：（1）通过矿物表面的自然催化合成单核苷酸；（2）在NR的催化下合成单核苷酸；（3）移动到临近的方格中。RNA分子可能发生的行为有：（1）模板介导的连接反应（磷酸二脂键的生成）分为自然连接和多聚酶介导下的连接；（2）双链分离：当模板的互补链不为空时，有一定概率两条链分离；（3）降解：包括单核昔酸降解成原料和RNA链断裂降解；（4）移动到临近的方格中。（5）含有特殊结构域的RNA分子作为核苷酸合成核酶催化该方格内的核苷酸前体形成核苷酸。系统中的每一种作用都假定有一定的发生概率。

本文基于 Ma的C语言模拟模型进行Java语言模型的改造与扩展。

2.2 模型模拟设置的参数及其定义

|  |  |  |  |
| --- | --- | --- | --- |
| 缩略语 | 英文全称 | 中文全称 | 设定值 |
| PSBP | Probability of separation of a base-pair | 碱基对分离的概率 | 0.5 |
| PBB | Probability of breaking of a phosphodiester bond | 磷酸二酯键断裂的概率 | 0.00001 |
| PLMC | Probability of ligation of two unit in a cell with mineral catalysis | 矿物催化细胞中的两个单元连接的概率 | 0.0000001 |
| PEL | Probability of formation of circular RNA from linear RNA via head-end ligation | 线状RNA头尾相连成环状RNA的概率 | 0.0000001 |
| FDA | Factor concerning de novo attraction of substrate by RNA template | 与RNA模板对底物从头吸引相关的因子 | 5 |
| FHR | Factor concerning self-cleavage or self-ligation of HR | 与HR自我剪切或自连接相关的因子 | 1 |
| FLT | Factor concerning linear RNA acting as template | 与线性RNA作为模板相关的因子 | 0.5 |
| PAT | Probability of attracting a substrate by a template when the substrate could base-pair with the template | 当底物可以与模板进行碱基互补配对时，模板能吸引到底物的概率 | 0.5 |
| PLT | Probability of a template-directed ligation | 模板定向连接的概率 | 0.2 |
| PMR | Probability of movement of raws | 原材料移动的概率 | 0.01 |
| PMV | Probability of movement of nucleotides | 核苷酸移动的概率 | 0.002 |
| PMF | Probability of mononucleotide formation from raw materials | 自然催化下原材料形成单核苷酸的概率 | 0.001 |
| PMFS | Probability of mononucleotide formation from raw materials under the catalysis of nt-synthetase | NR催化下原材料形成单核苷酸的概率 | 0.9 |
| PMD | Probability of mononucleotide decay to raw materials | 单核苷酸解聚成原材料的概率 | 0.05 |
| PNDE | Probability of a nucleotide residue decaying at RNA’s chain end | RNA链末端的核苷酸参即降解成原材料的概率 | 0.001 |
| PFP | Probability of false base-pairing | 碱基错误配对的概率 | 0.01 |
| RMRW | The relationship between the movement of RNAs and their molecular weight | RNA的移动概率与其分子重量之间的关系 | pow(p.length1 + p.length2, 1 / 2.0) |
| TNSS | Turn of nt-synthesis by nt-synthetase each step | NR一次能催化原材料的数量 | 1 |

最有效的仿真参数设定方法应该是根据实际情况来设定参数值。然而如前文所述，模型中的参数（如核苷酸衰变的概率和磷酸二酯键断裂的概率）可能会受到许多因素（如温度、pH值、金属离子浓度等）的影响，而我们目前对原始地球上的生化环境条件知之甚少，因此这些因素还不能设置到精确值。更实际的方法是，首先根据各行为发生概率的相对大小设置估计值，然后通过测试模型实现NR的成功繁荣（即NR出现在系统中并复制出显著多于其他种类RNA分子的数量规模）。接着我们就可以探索某一参数的变化是如何影响NR的繁荣的，据此推断NR出现的条件，未来当我们对原始地球的条件有了更多的了解时，就可以评估这一系列参数的可靠性。

重要的几个参数关系是：PMFS应该远大于PMF，以体现NR的催化高效性；PLT应该远大于PLMC（why），PMD可能略大于PMF（？why），PBB可能与PLMC相似数量级。此外，PFP是一个与复制保真度相关的参数。由于非酶复制往往不具有很高的保真度，因此PFP不应太小。为了实现NR的成功繁荣，通过后续的测试模拟，我们确定了参数的相对大小（表1）。需要注明的是，确定大小只是为了使参数的设置基本合理，并通过缩小范围来方便后续的分析。

该系统可被构想为一块岩石表面铺着一层薄薄的流体层。模拟的目的是观察NR是否可以在系统中扩散并成为一个具有显著规模的恒定种群，以及模型中的各参数变化如何影响结果。然后，NR的恒定种群可能为系统中进一步发生的达尔文进化提供基础。

2.3 模拟流程

开始

系统初始化NxN的二维方格

添加核苷酸前体至随机方格

是

结束

保存

是否已进行M次循环束

下一轮模拟

否

核苷酸前体随机发生事件

自然催化

成核苷酸

移动

是否有格子未进行模拟

否



随机选取还未模拟的格子

否

是否有RNA未进行模拟

下一个单元

是

RNA随机发生事件

移动

模板指导

下的连接

NR催化

双链分开

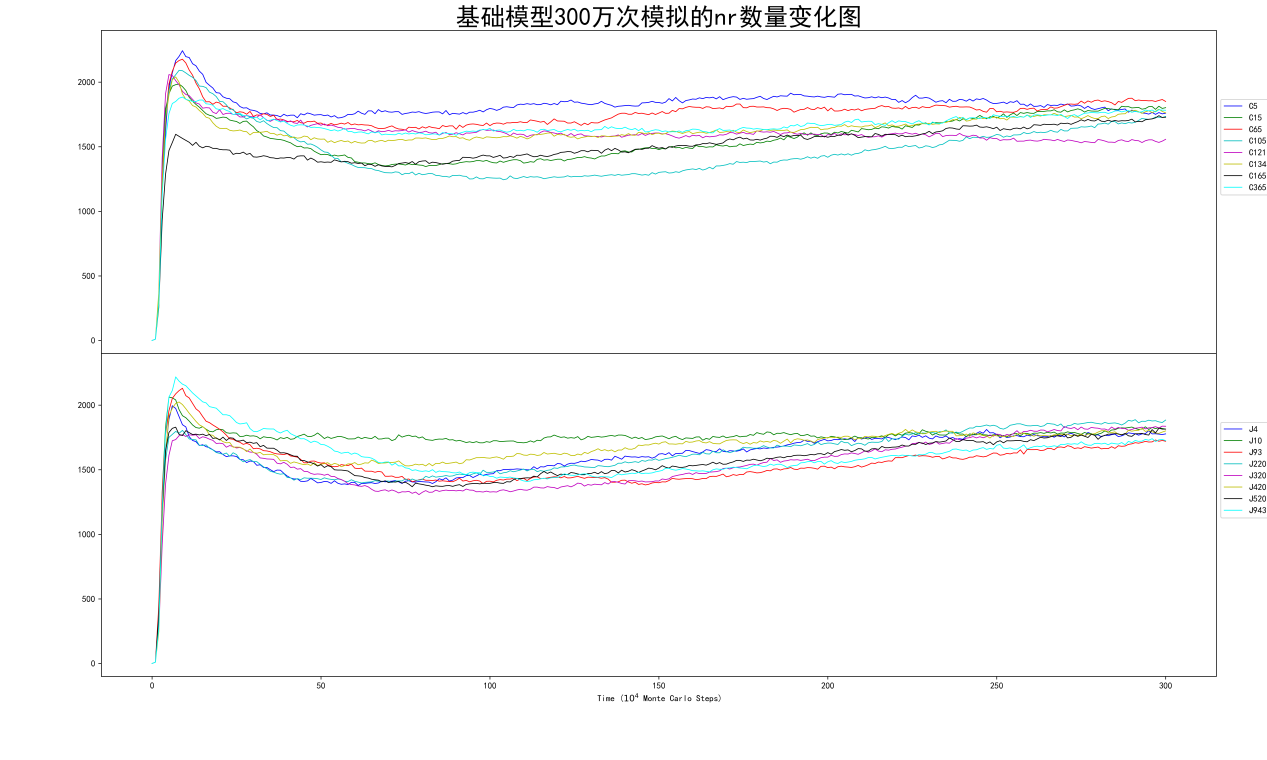
降解断裂

随机连接

3 实验结果

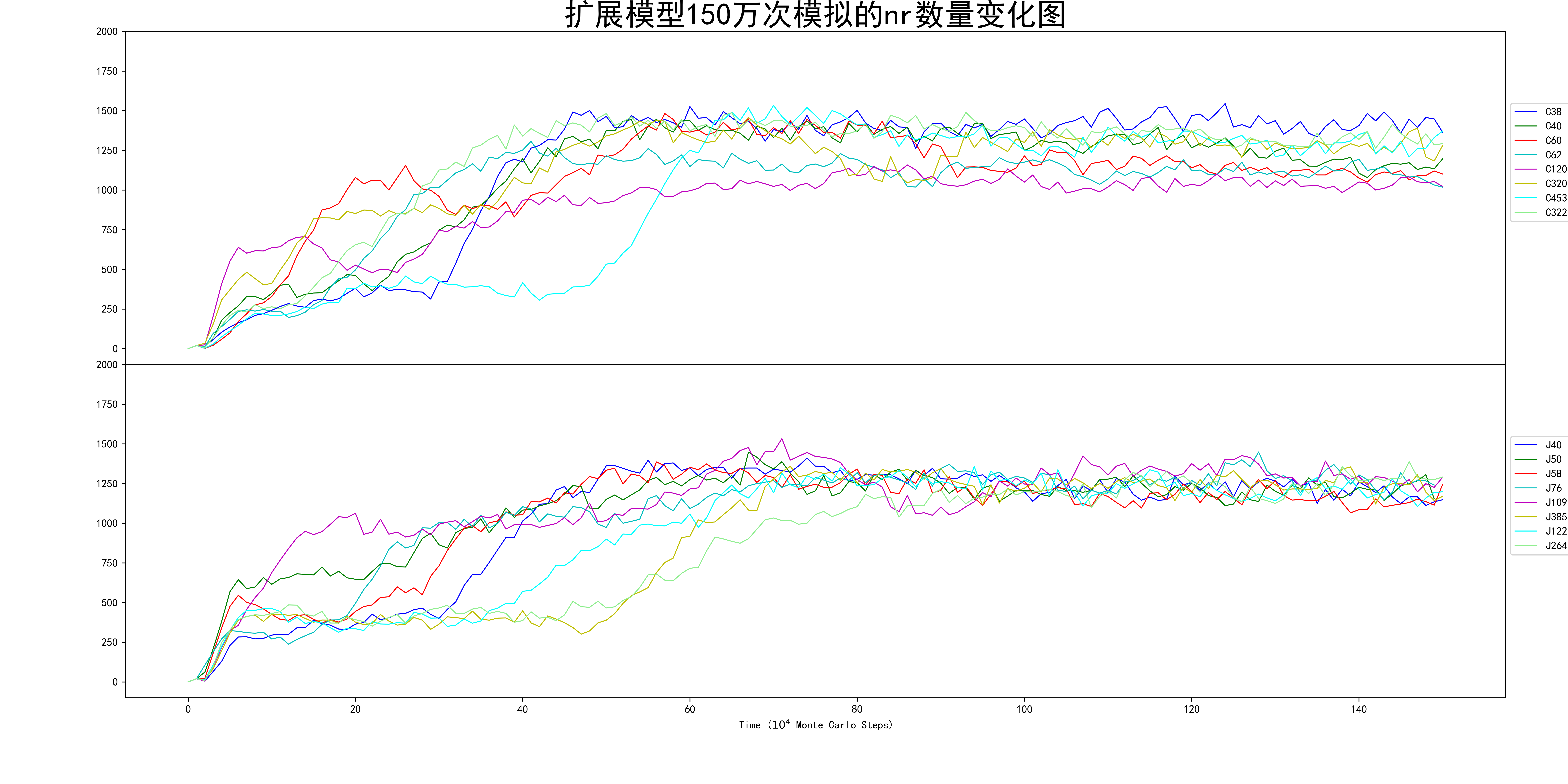
3.1 模拟结果

3.1.1 基础模型的模拟结果



主要参数：SIDE=30, TOTAL\_MATERIAL=40000, PSBP = 0.9, PBB = 0.000001, PLMC = 0.000002, PEL = 0.000002, PAT = 0.1, PLT = 0.9, PMOVR = 0.01, PMOV = (PMOVR/2), PMF = 0.0002, PMFS = 0.9, TNSS = 1, PMD = 0.01, PNDE = 0.001, PFP = 0.001, PNDE = 0.00002, INOCUNUM = 10.

3.1.2 扩展模型的模拟结果



主要参数：SIDE=30, TOTAL\_MATERIAL=80000, PSBP = 0.5, PBB = 0.00001, PLMC = 0.0000001, PEL = 0.0000001, MINI\_CIRCLE\_LENGTH = 8,

FDA = 5, FHR = 1, FLT = 0.5, PAT = 0.5, PLT = 0.2, PMR = 0.01, PMV = 0.002, PMF = 0.001, PMFS = 0.9, TNSS = 1, PMD = 0.05, PNDE = 0.001, PFP = 0.01, INOCUNUM = 20.

3.2运行效率对比

3.2.1 基础模型对比

|  |  |  |  |
| --- | --- | --- | --- |
| C语言种子值 | C语言运行时间（秒） | Java语言种子值 | Java语言运行时间（秒） |
| 555 | 978 | 210 | 3370 |
| 500 | 1164 | 943 | 2954 |
| 134 | 1065 | 320 | 2799 |
| 121 | 943 | 420 | 2804 |
| 105 | 1250 | 520 | 2826 |
| 165 | 995 | 4 | 2559 |
| 5 | 993 | 93 | 3157 |
| 65 | 919 | 10 | 3358 |
| 平均运行时间 | 1038 | 平均运行时间 | 2978 |
| 运行时间百分比：286% | | | |

3.2.2 扩展模型对比

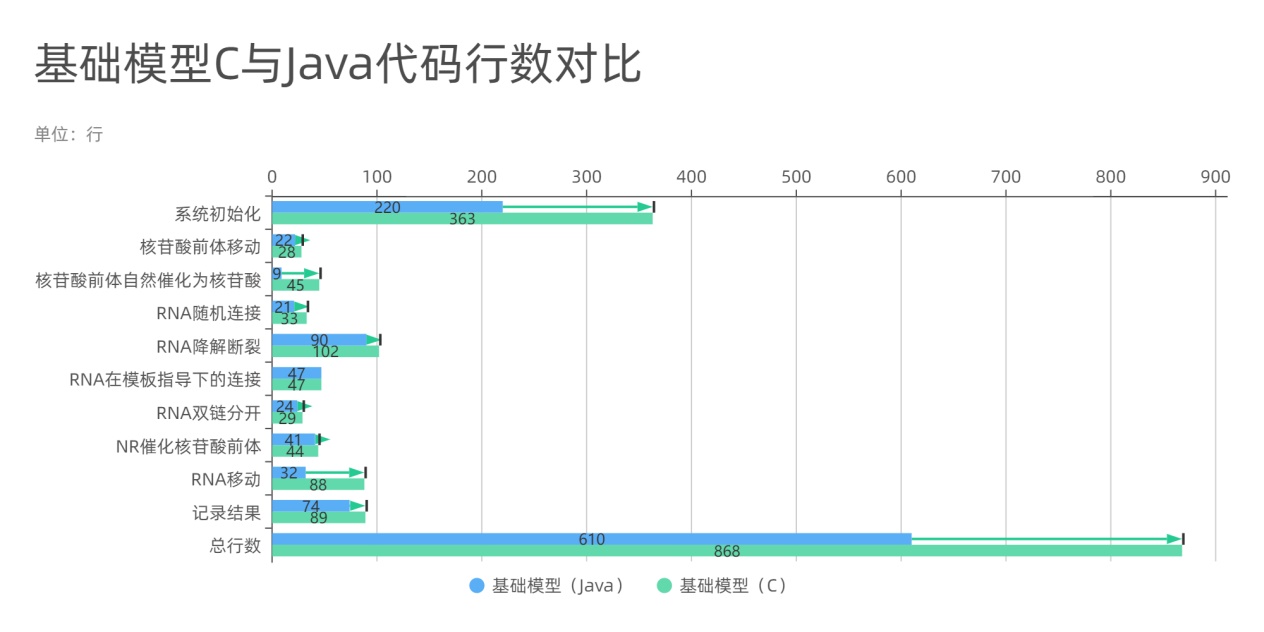
|  |  |  |  |
| --- | --- | --- | --- |
| C语言种子值 | C语言运行时间（秒） | Java语言种子值 | Java语言运行时间（秒） |
| 40 | 2531 | 264 | 4856 |
| 420 | 2583 | 50 | 5828 |
| 60 | 2186 | 48 | 4807 |
| 120 | 2206 | 109 | 5565 |
| 320 | 2535 | 385 | 4832 |
| 322 | 2533 | 76 | 5876 |
| 453 | 2725 | 121 | 4075 |
| 38 | 3737 | 122 | 4537 |
| 平均运行时间 | 2563 | 平均运行时间 | 5047 |
| 运行时间百分比：197% | | | |

3.3开发效率对比

3.3.1 基础模型对比

3.3.1.1 代码行数对比

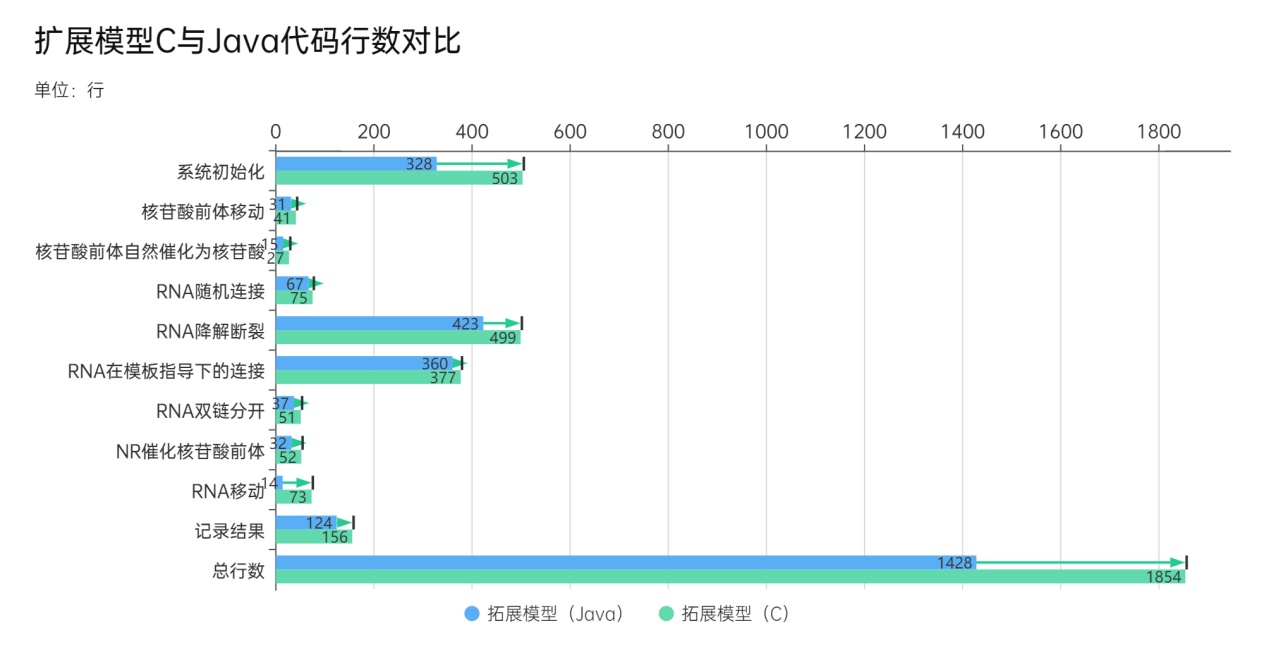
|  |  |  |  |
| --- | --- | --- | --- |
|  | 基础模型（C） | 基础模型（Java） | 简化百分比 |
| 系统初始化 | 363 | 220 | 39.39% |
| 核苷酸前体移动 | 28 | 22 | 21.43% |
| 核苷酸前体自然催化为核苷酸 | 45 | 39 | 13.33% |
| RNA随机连接 | 33 | 21 | 36.36% |
| RNA降解断裂 | 102 | 90 | 11.76% |
| RNA在模板指导下的连接 | 47 | 47 | 0.00% |
| RNA双链分开 | 29 | 24 | 17.24% |
| NR催化核苷酸前体 | 44 | 41 | 6.82% |
| RNA移动 | 88 | 32 | 63.64% |
| 记录结果 | 89 | 74 | 16.85% |
| 总行数 | 868 | 610 | 29.72% |



3.3.2 扩展模型对比

3.3.2.1 代码行数对比

|  |  |  |  |
| --- | --- | --- | --- |
|  | 拓展模型（C） | 拓展模型（Java） | 简化百分比 |
| 系统初始化 | 503 | 328 | 34.79% |
| 核苷酸前体移动 | 41 | 31 | 24.39% |
| 核苷酸前体自然催化为核苷酸 | 27 | 15 | 44.44% |
| RNA随机连接 | 75 | 67 | 10.67% |
| RNA降解断裂 | 499 | 423 | 15.23% |
| RNA在模板指导下的连接 | 377 | 360 | 4.51% |
| RNA双链分开 | 51 | 37 | 27.45% |
| NR催化核苷酸前体 | 52 | 32 | 38.46% |
| RNA移动 | 73 | 14 | 80.82% |
| 记录结果 | 156 | 124 | 20.51% |
| 总行数 | 1854 | 1428 | 22.98% |



4 讨论

4.1 结果讨论

从基础模型和扩展模型的C语言及Java语言版本的随机八个种子值的模拟结果来看，NR的繁荣具有普遍性且具有共同数量变化趋势：在前期数量剧增后逐渐趋于平稳。也从侧面证明Java语言模型与C语言模型具有相同的生命起源模拟效果。后续引入更复杂的因子，如能形成细胞膜的两亲分子后，NR的数量很可能会继续提升并逐渐形成进一步进化优势。

生命起源模拟模型的运行效率除了与语言本身运行机制、代码编写质量、模型复杂程度有关外，也与运行机器自身性能及当前工作状况相关，即使是相同模型用相同种子值在相同机器上运行，也无法保证运行时间完全相同，因此在对比运行效率时，本文采用了多次单机运行单个程序并统计运行时间取平均值的方法。

从结果分析，Java语言的基础模型运行时间是C语言的2.81倍，这是因其语言自身特点所决定的，但扩展后的模型运行时间是C语言的1.97倍，呈下降趋势。

在代码行数对比上，可以观察到无论是基础模型还是扩展模型，每一种功能的编写代码行数，Java语言均少于C语言，简化百分比可达到20%~30%。

4.1 java模型比C语言模型运行效率低的原因

Java语言和C语言作为高级语言，均需要进行编译才能运行，这个过程需要使用编译器，编译器的作用是将源代码转换成计算机能够识别的二进制形式。

Java语言和C语言的对比实际上是JIT编译器与静态编译器的对比。JIT编译器（Just-In-Time Compiler），即时编译器，是Java语言使用的编译器。它能够将源代码编译成为各种不同的机器代码，以适应对应的系统平台，最终使得程序在目标系统中得到顺利地运行。静态编译器是C语言使用的编译器。静态编译器在编译可执行文件的时候，将可执行文件需要调用的对应动态链接库中的部分提取出来，链接到可执行文件中去，使可执行文件在运行的时候不依赖于动态链接库。

Java语言编写的模型运行效率低于C语言编写的模型的原因是：

1.JIT编译器运行占用的是用户程序运行时间，而静态编译的时间成本在源码编译过程中，不在用户运行过程中。

2.Java语言是动态的类型安全语言，这意味着需要由JVM（Java虚拟机）来确保程序不会违反语言语义或访问非结构化内存。在实现层面上看，这就意味着JVM必须频繁进行动态检查，如对象实例访问时检查空指针、数组元素访问时检查上下界范围等。对于这类程序代码没有明确写出的检查行为，尽管编译器会努力进行优化，但是总体上仍然要消耗着不少的运行时间。

3. Java语言中的对象内存分配都是在堆上进行，只有方法中的局部变量才在栈上分配。而C语言的对象则有多种内存分配方式，既可以在堆上分配，也可以在栈上分配。

4.C语言中主要由用户程序代码free()来回收分配的内存，这就不存在无用对象筛选的过程，因此在运行效率上也比Java的垃圾收集机制要高。

4.2 java模型比C语言模型开发效率高的原因

1.二者的内存管理机制不同。C需要完全控制内存管理，Java主要依靠JVM的GC功能。

C语言在开发过程中需要开发人员对内存进行编程控制管理。C语言在语法限制以及变量的类型上比较宽松，这样就会影响到程序的安全性。

Java可以对内存中产生的垃圾进行自动收集，大幅度降低了程序的复杂程度。在开发过程中主要依靠JVM的GC功能进行内存管理，降低了开发人员手动编程控制管理。

2.C语言是面向过程的开发语言，而Java是面向对象的开发语言。面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用执行。

面向过程编程的优缺点：

- 开发过程中对类调用需要实例化，开销比较大，比较消耗资源。

- 面向过程是符合机器的思想

- 不易维护、不易复用、不易扩展。

- 是具体化的，流程化的，解决一个问题，需要一步一步的分析，一步一步的实现。

面向对象（Object Oriented Programming ，简称OOP）是一种编程范式，它使用“对象”来设计软件和数据结构。在面向对象编程中，对象是包含数据和能够对数据进行操作的方法的实体。有时对象也称作类。这种方法强调了数据的封装、继承和多态性，使得软件设计更加模块化，易于理解和维护。将现实的事物抽象出来，注意抽象这个词是重点啊，把现实生活的事物以及关系，抽象成类，通过继承，实现，组合的方式把万事万物都给容纳了。实现了对现实世界的抽象和数学建模。这是一次飞跃性的进步。

面向对象编程的核心概念包括：

- 封装：隐藏对象的内部状态和实现细节，仅对外提供公共的访问方式，将变化隔离，便于使用，提高复用性和安全性。

- 继承：允许新创建的类（子类）继承现有类（父类）的属性和方法。提高代码复用性

- 多态：允许不同类的对象对同一消息做出响应，即同一操作可以作用于不同的对象上，产生不同的结果。

面向对象编程的优点是：

- 提高了代码的复用性、可维护性和可扩展性。特别是在需要大量代码重用和模块化设计的情况。

- 可以设计出低耦合的系统，使系统更加灵活、更加易于维护

- 抽象会使复杂的问题更加简单化，它支持更高级的抽象和更容易的代码管理。

- 面向对象更符合人类的思维。

- 是模型化的，只需要抽象出一个类，像一个封闭的盒子，在这里你拥有数据也拥有解决问题的方法。需要什么功能直接使用旧可以了，不必去一步一步的实现，至于这个功能是如何实现的，不需要去管，会用就可以了。

你可以理解为面向对象编程最终的目的是对要解决的问题进行问题的抽象和数学建模。利用模型去解决问题。

或者可以理解为面向对象的底层其实就是面向对象，把面向对象抽象成类，然后封装，方便我们使用，这就是面向对象了。

多态和继承是相辅相成的。举例说明：

打印机，包含黑白打印机、彩色打印机。

此时，打印机它属于一个父类，黑白打印机属于子类，彩色打印机也属于子类

打印机的关系

1. 继承：黑白打印机和彩色打印机都继承自打印机这个父类。例如打印机父类有一个自动翻页打印功能，那么黑白打印机和彩色打印机就自动拥有了自动翻页打印功能。

2. 重写：使用打印机进行打印（此时打印机有一个打印方法），如果打印机为彩色打印机，则打印出来的就是彩色的（此时彩色打印机需要重写打印机的打印方法，让其打印效果为彩色）；如打印机为黑白打印机，则打印出来的就是黑白的（此时黑白打印机需要重写打印机的打印方法，让其打印效果为黑白的）。

3. 引用：new一个打印机对象时，父类打印机的引用可以指向具体的子类对象（彩色打印机或黑白打印机）。Print parent = new ColorPrint();父类打印机的引用指向子类彩色打印机。

5 小结与展望

优化算法的可能？