



Google 面向教育者的计算思维课程

课程目录

目录

一、计算思维简介.....	4
1-1 什么是计算思维?	4
1-1-1 计算思维 (CT: Computational Thinking)	5
1-1-2 计算思维要素	6
1-1-3 将计算思维应用在未来课堂中	6
1-2 课程如何运作.....	8
1-2-1 课程结构.....	8
1-2-2 其他内容	9
二、算法探究.....	10
2-1 概述.....	10
2-1-1 计算机科学: 旅行.....	11
2-1-2 人文学科: 与时俱进的字词.....	14
2-1-3 数学: 元胞自动机	17
2-1-4 科学: 基因组学.....	21
2-2 课程反思.....	24
三、模式发掘.....	26
3-1 概述.....	26
3-1-1 计算机科学: 数据压缩.....	27
3-1-2 人文学科: 音乐.....	32
3-1-3 数学: 小海龟几何	35
3-1-4 科学: 分类.....	40
3-2 课程反思.....	44
四、算法开发.....	46
4-1 概述.....	46
4-1-1 计算机科学: 汉诺塔.....	47
4-1-2 人文学科: 聊天机器人	51
4-1-3 数学: 计算器	54
4-1-4 科学: 弹力球	60
4-2 课程反思.....	63

五、应用计算思维.....	66
5-1 项目概要.....	66
5-2 反馈、评估、评分	66
5-2-1 项目，第一部分.....	66
5-2-2 项目，第二部分.....	66
5-3 样例项目	67
5-4 总结	68

课程简介

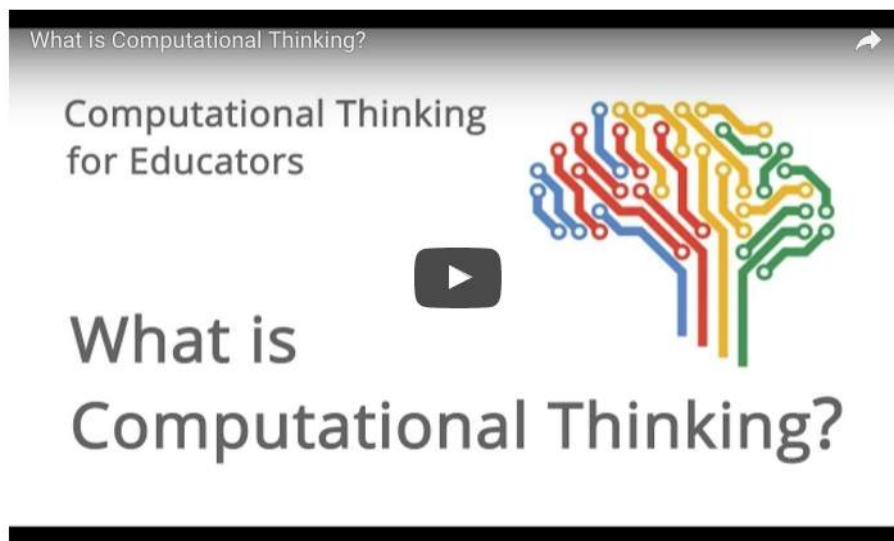
本课程的目标是帮助教育工作者学习计算思维（CT: Computational Thinking），了解它与计算机科学的区别，以及理解如何将其整合到不同的学科中。在课程学习过程中，你将深化对于计算思维的认知，探究计算思维与特定学科相结合的案例，参与将计算思维应用到特定学科领域的综合实践活动，制定一个计划将计算思维整合到你自己的课程中。

本课程分为五个单元，每个单元的要点如下：

- 计算思维简介：什么是计算思维，它出现在哪里？它为何需要你的关注？以及它是被如何应用的？
- 算法探究：课程带你亲历学科领域中的算法实例。认识到算法是一种可以提高学习者做事能力的强大工具，同时认识到技术对于实现算法和完成算法的自动化是非常有用的。
- 模式发掘：探索学科案例中蕴含的模式例子，通过模式识别形成一套自己独有的、探究问题的流程。
- 算法开发：增强你在解决问题过程中应用算法过程的信心，并了解算法是如何清晰地表达一个过程或规则。
- 课程项目：应用计算思维。撰写一份如何将计算思维应用到你的学科中的计划，陈述如何将计算思维整合到你的日常工作和课堂教学中。

一、计算思维简介

1-1 什么是计算思维？



这部分是 5 分 38 秒的视频，译文如下：

如何绘制人类的完整 DNA 序列？威廉莎士比亚的著作是否全部亲笔所著？是否能编写出可自主作曲的智能电脑程序？以上这三个现实问题有什么共性吗？要想回答这些问题，需要使用所谓的计算思维。每天都有越来越多的人正在使用这种思维方式解决各式各样的问题。因此，无论教授数学、科学、计算机或者人文学科，计算思维都能为你的课堂活动添砖加瓦。那么，什么是计算思维呢？

计算思维是一种问题解决的方式。这种思维将问题分解，并且利用所掌握的计算知识找出解决问题的办法。计算思维可以划分为四个主要组成部分：其一，所谓“解构或分解”，即把问题进行拆分，同时厘清各个部分的属性，明晰如何拆解一个任务；其二，所谓“模式识别”，即找出拆分后问题各部分之间的异同，为后续的预测提供依据；其三，所谓“模式归纳”，或“抽象化”，即探寻形成这些模式背后的一般规律；其四，所谓“算法开发”，即针对相似的问题提供逐步的解决办法。

如今人们尝试在许多学科领域中应用计算思维解决问题。当人们提出易被计算机解决或者通过大数据分析探寻内部规律的难题的时候，表明他们正在运用计算思维进行思考。计算思维带动了计算生物学、计算化学等领域的发展，同时也带来了能够运用在文学、社会研究和艺术方面的全新技术。计算思维早已来到我们身边，存在于我们生活各处，所以我们没必要对其感到恐惧。

有些人将计算思维和计算机科学混为一谈，但事实上这二者泾渭分明。计算机科学是一门学术性学科，包括了可计算性的研究及其借助计算机的应用。相比之下，计算思维则是我们攻克难题的一种方式。即是一种直观与抽象的思维方式。无论你是否是计算机科学家，还是其他什么人，你都会用到计算思维。这是一种让人获益匪浅的普适思维技能。计算思维还能增加学生面对模糊的、复杂的或开放性的问题时候的自信心。因此，任何学科背景或年龄段的学生都应该学习计算思维。

将问题分解成大小不同的部分，并逐一处理解决，最终进行总结归纳以解决整体问题，这是一种让人受益的技能。假设你现在是一名记者，想要了解人们对待即将到来的大选所持的各种观点。一种方法是在大街上询问他们的想法，另一种方法是使用一种能够在几秒之内分析数以万计的社交媒体网站内容的计算机程序。这是一个能够体现基于计算思维解决问题优势的例子。因此，过去的学生如果想要寻找和理解数据，需要人工收集数据，而现在的学生可以通过改进算法迅速地获得结果，将更多的注意力放在对数据进行深入研究上，而不是放在收集与计算数据上面。通过运用科技的力量，我们能够会聚精力，增强洞察力，分析具体情况，并且得出结论。

学生们不应该只是简单地使用教给他们的模型，而应该能够创建出来自己的模型。为了做到这点，学生们必须借助计算思维，发现或者洞察出隐藏在事物背后的联系。计算思维不是额外的负担或需要单独教授的内容，而是对现有课程的优化。因此，早期我多以日常生活的案例来说明计算思维，帮助学生理解计算思维是一种与其自身息息相关的事物，他们就会明白计算思维确实存在于生活的各个方面。如果我们的学生掌握了计算思维技能，成为技术创新者，那么，他们不仅能够在国际专业领域找到立足之地，还可以通过创造性地解决重要问题而造福社会。

那么如何绘制人类基因序列呢？答案是借助算法与电脑程序给 DNA 中数以百万计的碱基对进行排序。如何破解莎士比亚著作之谜呢？答案是通过计算机分析莎翁作品的词汇、主题和风格，能够确认莎士比亚确实编著了自己名下所有的作品，实至名归。至于那个如何实现智能作曲的问题，则可以通过计算思维发现已有音乐作品中的存在方式与规律，编写程序，生成全新的音乐作品。今天的人类所面临的全球重大问题，都需要跨学科来解决。通过将计算思维的技能整合进入所有学科之中，所培养出来的学生能够对过去看似无解的问题，提出全新的解决方案。当然今天或许还无法解决很多大自然的谜题，诸如重力如何产生，能否治愈癌症，等等，但是在不久的将来，我们的学生或许能够借助计算思维的技巧回答这些问题。

1-1-1 计算思维（CT: Computational Thinking）

计算思维是一个有着诸多特点和要求的问题解决过程。计算思维不但对于计算机应用的发展非

常重要，而且它也可以用于所有学科的问题解决，包括人文学科、数学和科学等。学生在完成计算思维的课后，不但能了解各学科之间的关联，同时也能够体会到课外生活与课堂内知识之间的联系。

1-1-2 计算思维要素

本课程提供了一次机会来体验计算思维的基本要素，包括：

- **分解：**把数据、过程或问题分解成更小的、易于管理或解决的部分
- **模式识别：**观察数据的模式、趋势和规律
- **抽象：**识别模式形成背后的一般原理
- **算法开发：**为解决某一类问题撰写一系列详细的指令。

国际教育技术协会（ISTE），计算机科学教师协会（CSTA）和英国学校计算课程工作小组（CAS）与教育界和工业界的代表们合作，共同开发了面向教育工作者的计算思维资源。

[ISTE 计算思维网站](#)

[CSTA 计算思维网站](#)

[学校计算课程工作小组（CAS）计算思维网站](#)

[谷歌探索计算思维（ECT）网站](#)（见附件一）

1-1-3 将计算思维应用在未来课堂中

我们的目标是你可以尽快地在教学中使用或传授计算思维相关概念。相比于重新创设一门计算思维的新课程而言，教师在自己已有的课堂教学中使用计算思维的方法和概念反而能够获得更好的效果。表 1-1 显示了计算思维与计算机科学的区别：

表 1-1 计算思维与计算机科学的区别

计算思维概念	计算机科学应用
把问题分解为若干部分或步骤	将计算图表问题拆分成四部分来处理，每部分由不同的计算机程序独立完成

识别并发现模式或趋势	可视化数据来比较微芯片材料和计算机速度，并注意到它的发展趋势
开发解决问题或任务步骤的指令	编写一个计算机程序来对数据进行排序
把模式和趋势归纳至规则、原理或见解中	相对于复杂的编程，用较少代码实现复数数据结构

计算机科学是关于信息的研究：如何表征信息？如何优化存储信息？如何处理信息？计算机科学是使用计算机进行计算及其应用的研究。而另一方面，计算思维则包括编写计算机程序时所使用的技能和思维方式。下面我们来看表 1-2。

表 1-2 计算思维在学科领域中的应用

计算思维概念	学科领域应用
把问题分解为若干部分或步骤	文学：通过对韵律、韵文、意象、结构、语气、措词与含义的分析来分析诗歌
识别并发现模式或趋势	经济：寻找国家经济增长和下降的循环模式
开发解决问题或任务步骤的指令	烹饪艺术：撰写供他人使用的菜谱
把模式和趋势归纳至规则、原理或见解中	数学：找出二阶多项式分解法则
	化学：找出化学键（类型）及（分子间）相互作用的规律

在左列中，所有技能都是计算思维涉及的技能或概念。而在右列，这些技能被应用到文学、经济、烹饪艺术和音乐中。就本质来说，计算思维是计算机科学家的基本技能和思维方式。然而你可以将它应用在你所教授的学科领域或主题，甚至是任何学科领域或主题。并且，你可以在设计流程或算法以解决问题过程中，随时应用这些思维技巧。

你可能会注意到课程中包含了程序设计的内容。计算思维并不囿于使用某一种编程语言，然而

编程其实是一种很好的观察问题解决方式的途径。编写代码既不是本课程的学习目标，也不是开始本课程学习的前提条件。如果你在课程中的某个环节遇到障碍，包括代码部分，可以在[课程社区](#)（或者参考附件二）中寻求帮助。

1-2 课程如何运作

这门课程的目的是增加世界范围内教育者对于**计算思维（CT: Computational Thinking）**的了解认识并鼓励他们将其整合到他们的教学大纲中。这门课程分为五个部分，其主要内容包括：

- **计算思维简介：什么是计算思维**——计算思维是什么，哪里可以使用计算思维？为什么我们要关注计算思维和如何应用计算思维？
- **算法探究**：回顾一下在你熟悉的领域中使用算法的例子。思考一下为什么算法是一个可以让你事半功倍的强大工具，各种技术如何借助算法执行以及自动化，提升效能。
- **模式发掘**：通过探索存在不同领域中不同事物中的各种模式，形成属于你自己的模式化解决问题的流程。
- **算法开发**：增加你对使用计算思维解决指定问题的信心，并且让你了解如何用算法表达问题解决过程或者规则。
- **课程项目**：应用计算思维。阐述你如何在学科中应用计算思维，并尝试制定一个将计算思维整合到你的工作与课堂的计划。

1-2-1 课程结构

课程：由五个单元组成，其中包括一个由两部分组成的课程项目

单元：由一组小课程组成；每个单元包含一组适合四个不同群体的混合课程与活动：

- 人文学科教师
- 数学教师
- 科学教师
- 计算机科学教师

课程活动：包括用于增强学习者计算思维意识的实例模拟、程序与练习，计算思维整合案例

的展示，并允许学习者通过互动方式将计算思维应用到各自的学科领域。课程活动同时提供了完成活动任务的操作步骤，延伸学习内容的一些链接，用于实践技能和反馈获取的活动，以及一个分享想法和寻求帮助的讨论社区。

课程项目：提供把你在课堂上所学技巧应用于实践的机会。

1-2-2 其他内容

更多的资源

本课程仅是开始，找到课程、资源、刊载论文，更多请参阅[探索计算思维](#)（或者参考附件一）。

讨论社区

这是你提出问题、分享创意，并向同行提供反馈的地方。

- 国内讨论社区：TBD
- Google 全球计算思维课程讨论社区：[计算思维社区](#)链接（或者参考附件二）

二、算法探究

2-1 概述

在本节中，你可以体验一些稍作修改就能用于你的课堂活动的活动。这些活动不需要任何编码，并打算提供一些合理使用算法和计算思维的例子，请点击下面的活动的链接来体验吧。也欢迎随意体验学科领域以外的活动，像你所期待的学生们那样探索、试验、游戏和享受吧！



图 2-1 旅行案例示意图

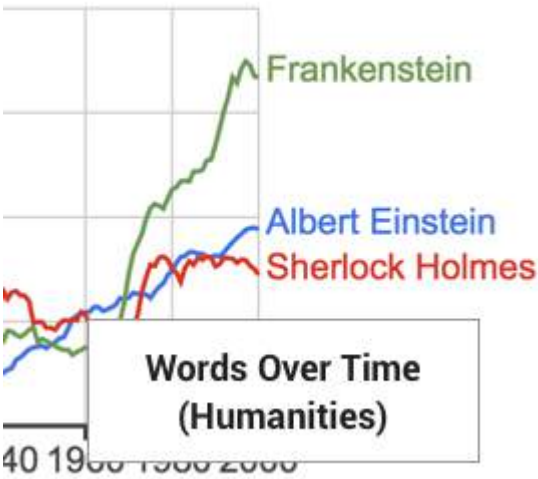


图 2-2 与时俱进的字词案例示意图

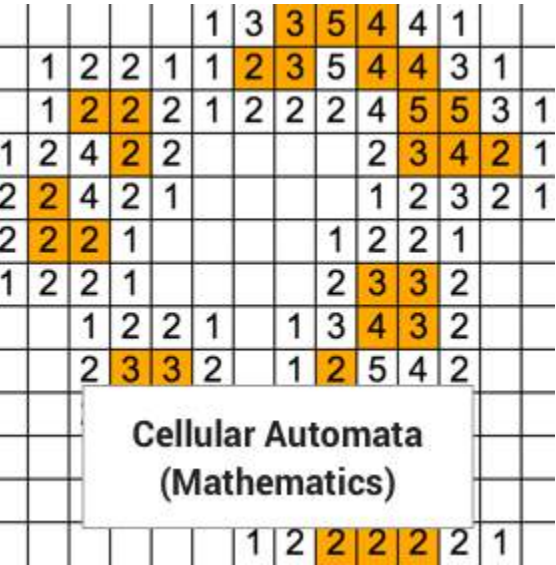


图 2-3 元胞自动机案例示意图

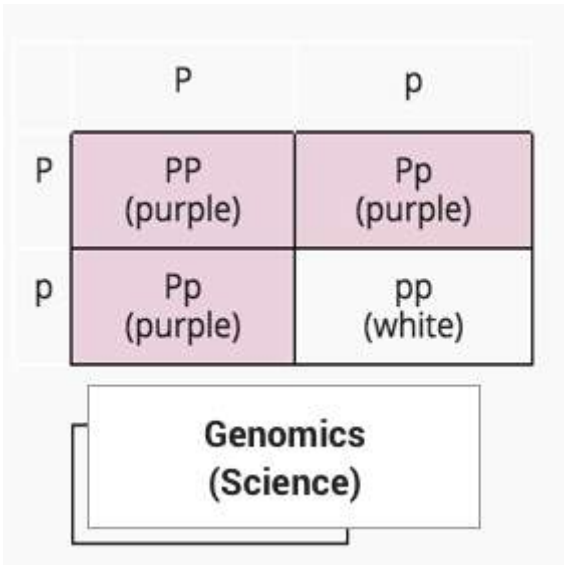


图 2-4 基因组学案例示意图

2-1-1 计算机科学：旅行

你和家人正在计划穿越区域内所有主要地点的自驾游，且你需要规划出最短的路线以节省燃油成本。

在图 2-5-a 中的地图中从下拉菜单中选择纬度、经度、随机、可拖拽，运行一个自驾游样例。

看到结果后，尝试将模式切换到“可拖拽”，并完善出你认为最好的路线。

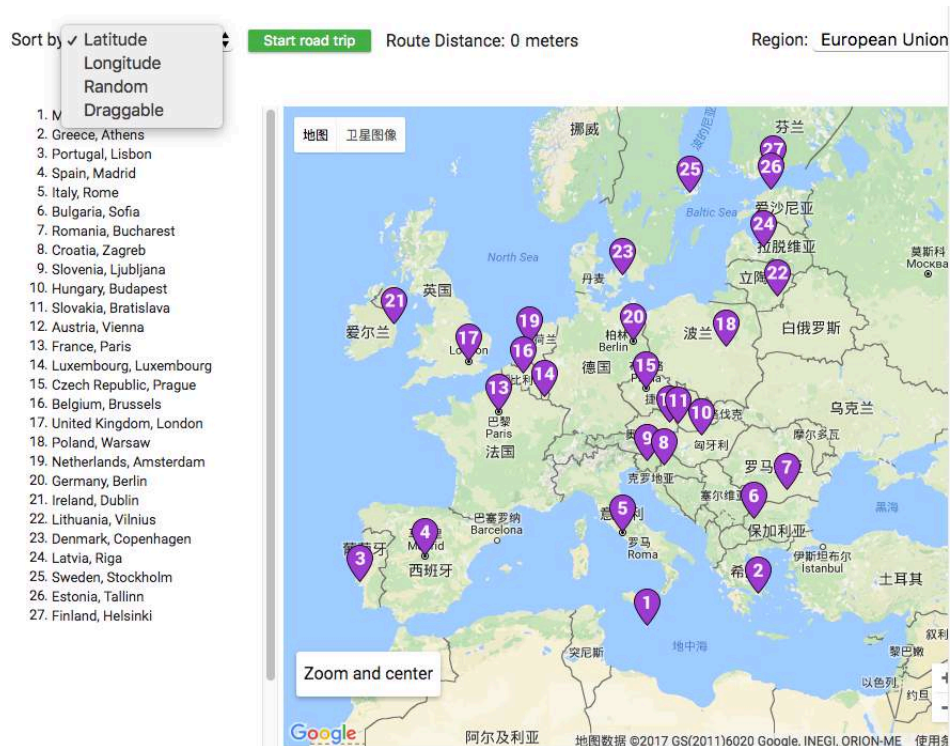


图 2-5-a 交互式地图探索示意图（所有国家）

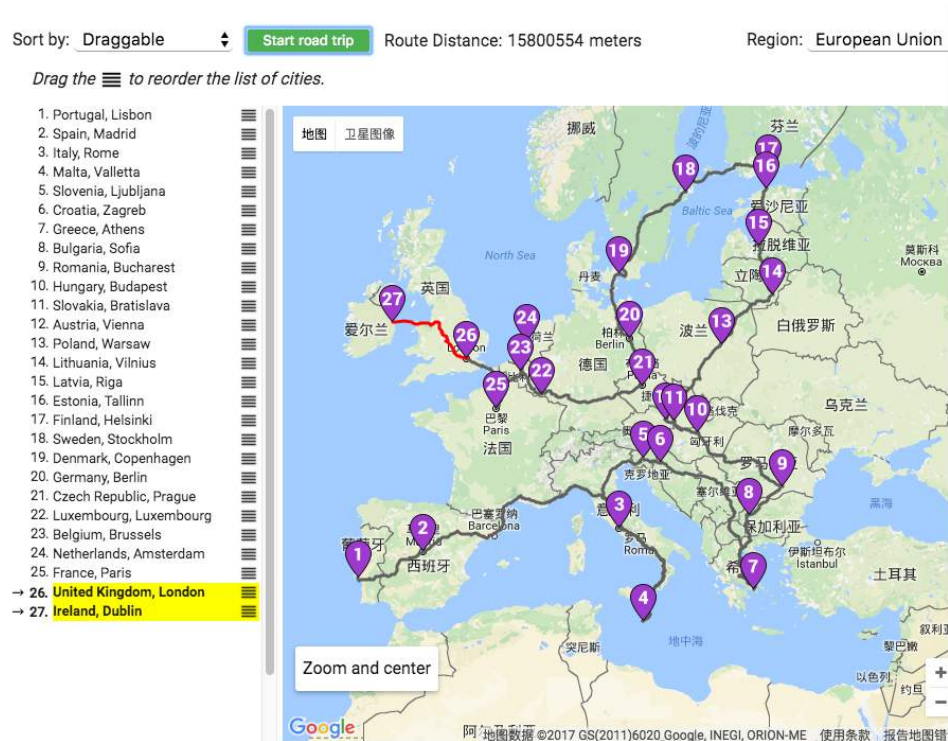


图 2-5-b 交互式地图探索示意图（可拖拽模式下的路线规划）

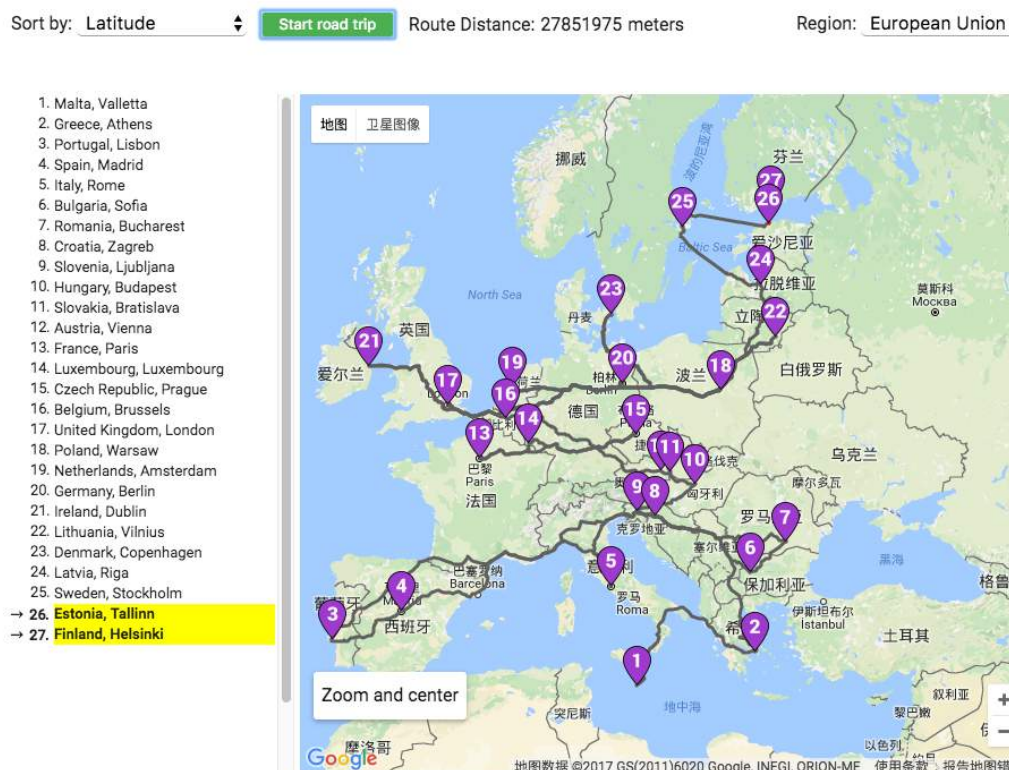


图 2-5-c 交互式地图探索示意图（按纬度排序模式下的路线规划）

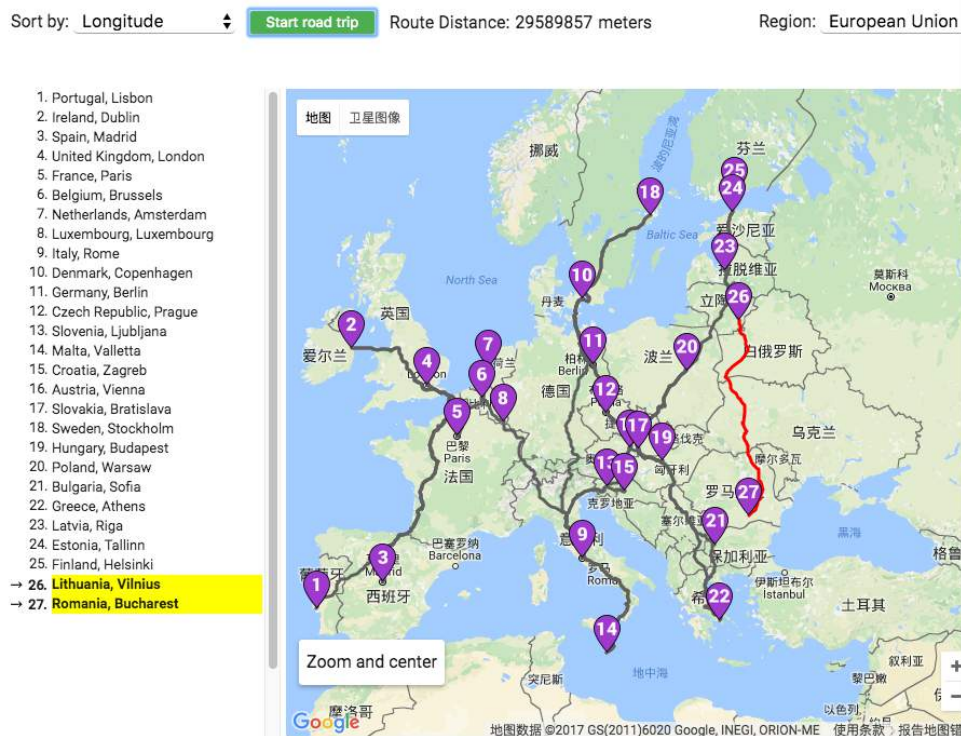


图 2-5-d 交互式地图探索示意图（按经度排序模式下的路线规划）

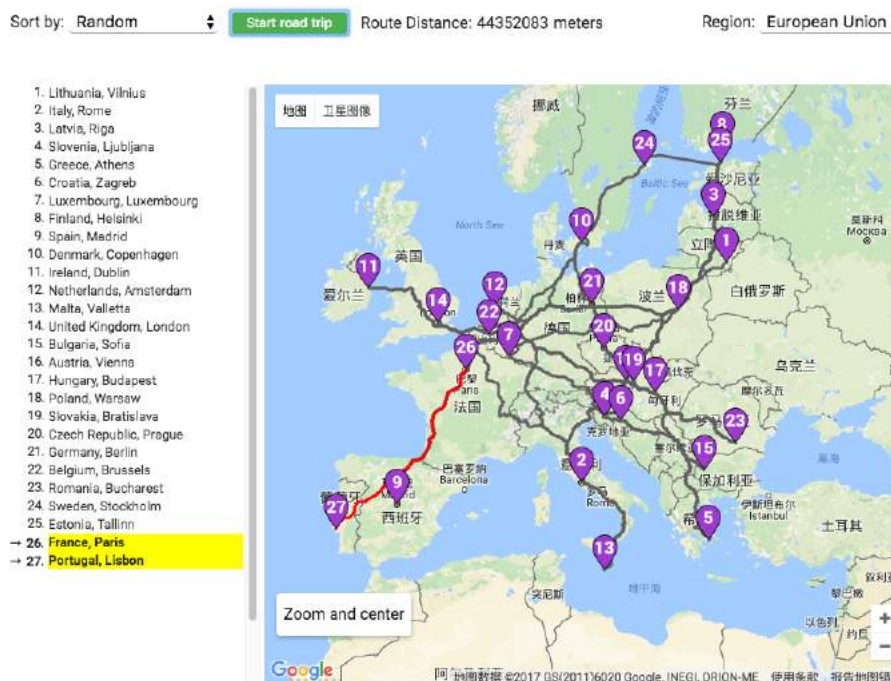


图 2-5-e 交互式地图探索示意图（随机模式下的路线规划）

互动时地图网址（原课程英文网站）

现在你有机会测试一些选项，并思考如何解释来说服家人采用你所建议的路线。你的提议中是

否包含一些城市列表或诸如“由西向东”的指导原则呢？

以下是一些需要考虑的问题：

1. 出发城市的选择是否重要？

2. 你的路线策略（经度、纬度等）适用于任何国家还是需要因地制宜？请尝试在另一个国家中使用你的策略，看它是否同样有效。

3. 这个难题的成因是什么？

这类问题通常被称为旅行推销员问题，其面临的挑战在于每个城市只能经过一次，并且最后需要回到起点城市。虽然此类问题尚未被人们完美地解决，但是人们研究这个问题和发展算法试图解决这个问题（即使以一个次优的方式）的过程与成果，已经被用来提高许多其他相关问题的效率。路径规划和海量数据搜索（如 DNA 和网页），就是两种类似的问题。

如希望进一步探讨此主题，请在互联网搜索：旅行推销员问题、NP（Non-Deterministic Polynomial, 非确定多项式）问题。若与学生一起使用，可为此活动增加活动标准（或者参考附件三）。

2-1-2 人文学科：与时俱进的字词

假设你想知道近些年在不同文化背景下，“爱”这个字是如何被使用的。假定书籍就承载了文化中有价值和有意义的部分，如果要清点“爱”字在一本书，甚至是大量书中出现的次数，要花费你多长时间？谷歌 Ngram Viewer（见图 2-6-a 所示）直观地显示出许多已经被谷歌扫描过的书籍里某个字或词的出现次数。

请尝试在下面的搜索栏里把“love”字替换为其它字词。随着时间的推移，哪些字词呈现上升或下降趋势？是否有字词在图表中呈波峰曲线？

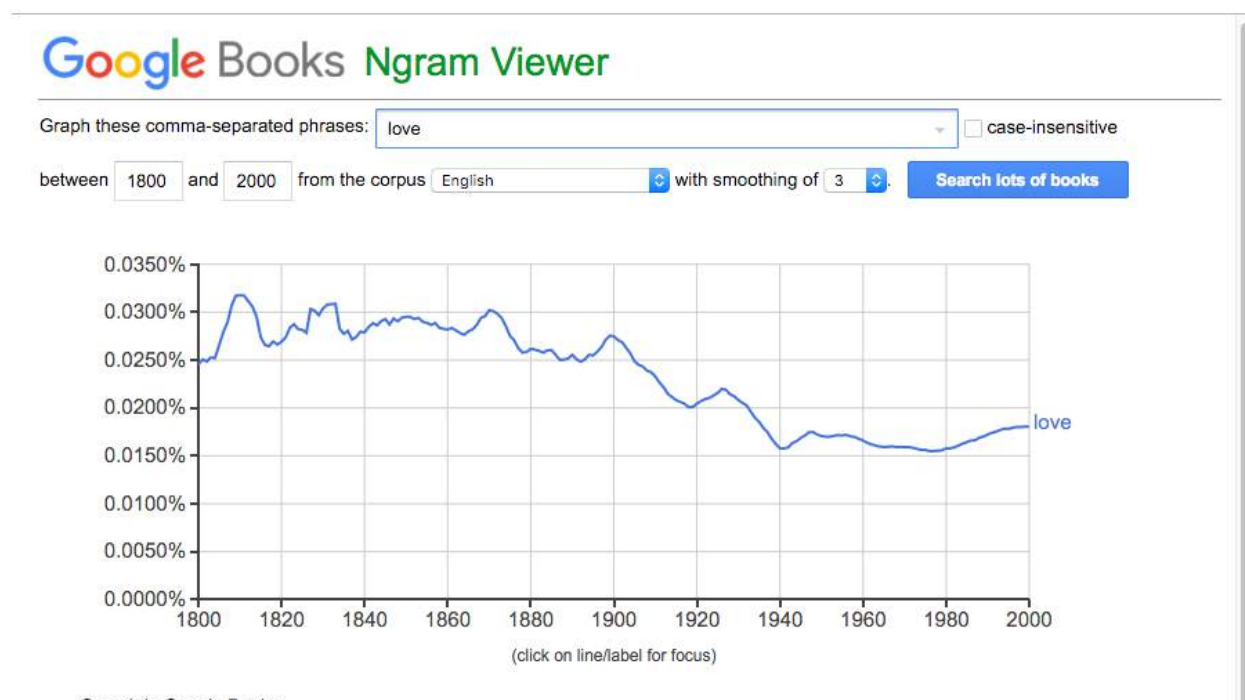


图 2-6-a 谷歌 Ngram Viewer 的搜索结果（关键词“love”）

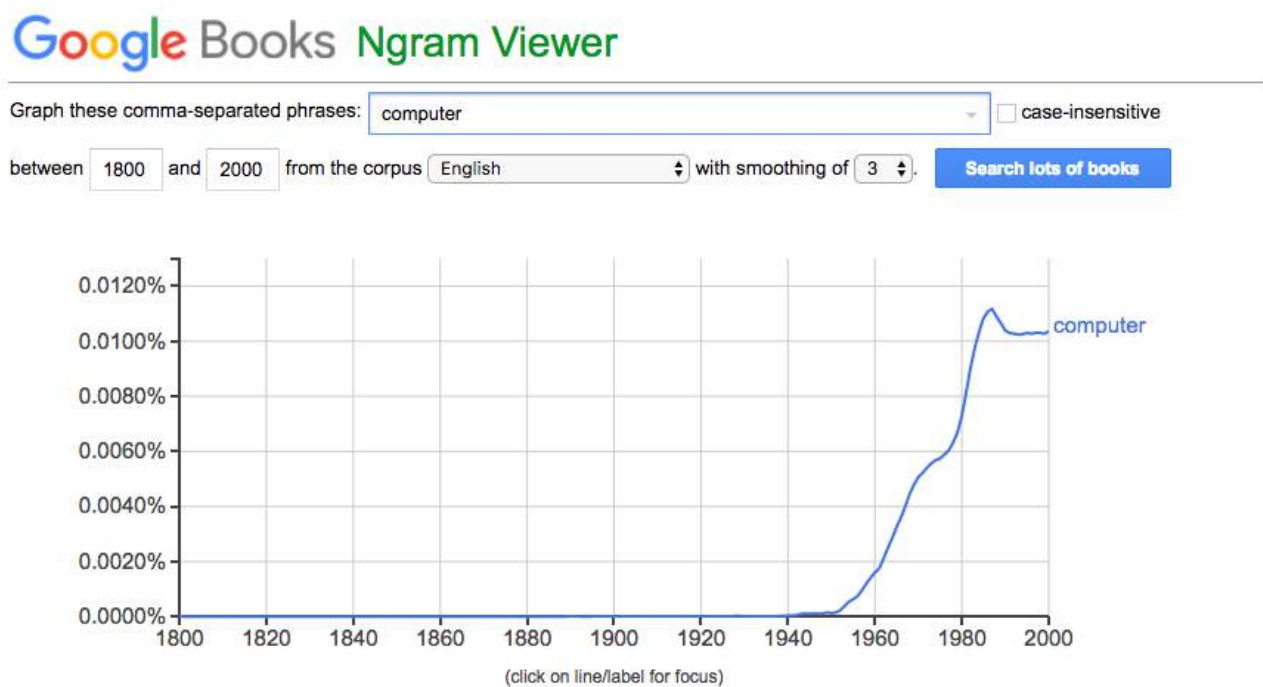


图 2-6-b 谷歌 Ngram Viewer 的搜索结果（关键词“computer”）

Google Books Ngram Viewer

Graph these comma-separated phrases: ☐ case-insensitive
between and from the corpus with smoothing of [Search lots of books](#)

Classical Chinese (before 1900) uses a vocabulary and grammar that differs significantly from modern Chinese.



图 2-6-c 谷歌 Ngram Viewer 的搜索结果（关键词“爱”）

Google Books Ngram Viewer

Graph these comma-separated phrases: ☐ case-insensitive
between and from the corpus with smoothing of [Search lots of books](#)

Classical Chinese (before 1900) uses a vocabulary and grammar that differs significantly from modern Chinese.



图 2-6-d 谷歌 Ngram Viewer 的搜索结果（关键词“计算机”）

互动网址：<https://books.google.com/ngrams>

你把今天的报纸交给一群人，并让他们找到出现频率最高的字，你建议他们用什么方法以便在最短时间内得出答案？

若要求同样一群人用几句话总结当天的新闻，你会怎样提示以协助他们找到当天最重要的信息？假若是回顾一年里报纸最有价值的信息又将如何呢？用于扫描书中所有字词的算法和抓取网页，从中分析重要的，和你的查询有密切关联语词的过程是类似的。此类算法已使搜索大量数据信

息成为可能。如果你想继续探索这些可能性，请尝试以下想法：

- 1. 在 Ngram Viewer 中查询多个单词。请键入“love”、“money”至上面 Ngram 的搜索栏，并对趋势是否是一个巧合、一个有意义事件的反映或文化变迁提出假设。加入各种情绪如“快乐”、“悲伤”，以查看这些趋势与你的搜索词是否相关。
- 2. 尝试相反的方法。针对一件特殊的重要事件，思考看能否找到恰当搜索关键词，可以显示在一段特定时间段内该事件提及次数不断增加。
- 3. 哪种媒介最能代表你的文化修养？报纸、书籍、音乐、电影、短信、YouTube 视频或其它？

在互联网上搜索“ngram”、“文化学”、“齐普夫定律”、“莎士比亚著作”和“谷歌的时代精神”可获知更多主题。如果学生使用，此活动的潜能标准（或者参考附件二）亦可作相应匹配。

2-1-3 数学：元胞自动机

现在有一个由多个方块组成的网格，当点击某个方块就可以激活它，方块的颜色也会从白色变为橙色，方块上的数字代表与它相邻的方块有多少是“存活”的（每个方块都有 8 个相邻的方块）。例如，如果只有一个方格被选中，然后所有方格周围显示 1。图示 2-7 是一些例子：

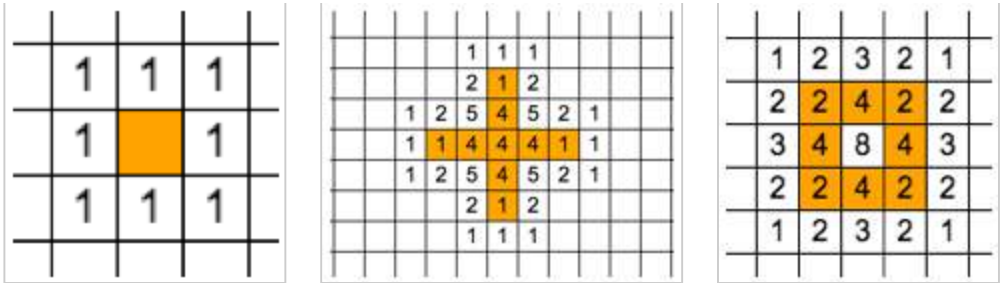


图 2-7 元胞自动机运行示例

请尝试点击网格中的一些方格，然后按几次前进，观察网格如何变化。在图 2-8 的运动界面中，你还可以单击开始/停止来看看它继续运行。是什么导致了方格的出现和消失？

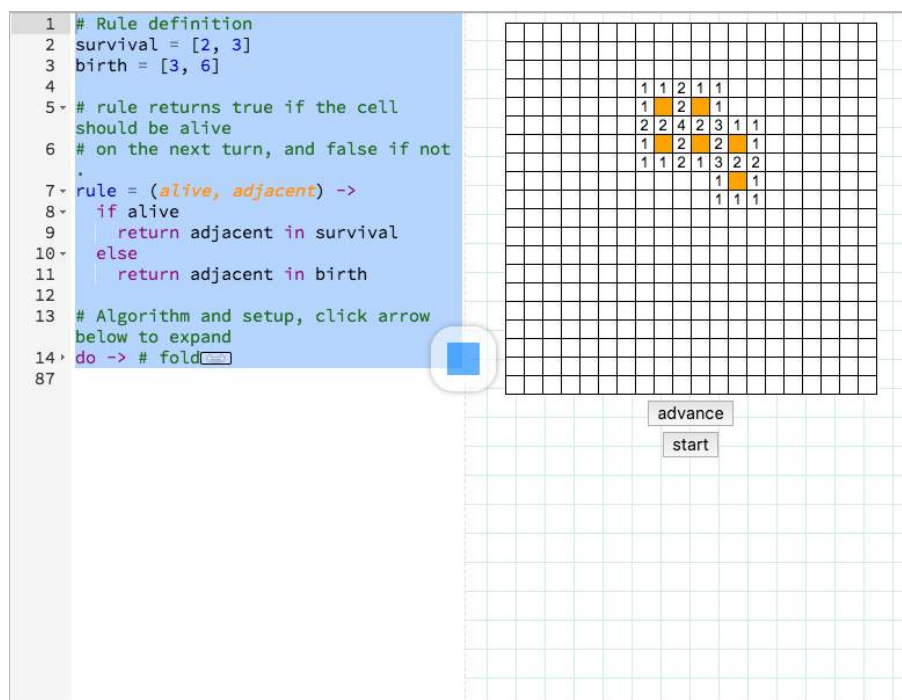


图 2-8 元胞自动机基于 pencilcode 的代码与运行界面

互动链接: <https://jsuansiwei.pencilcode.net/edit/MathCellularAutomata>

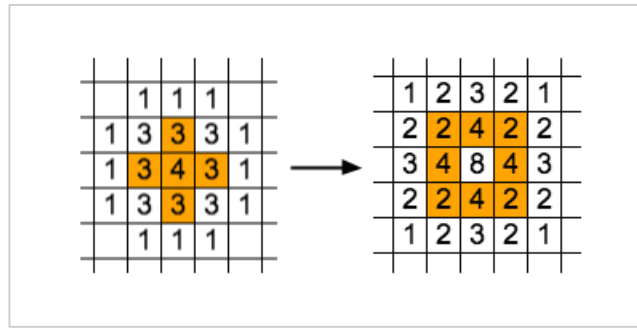
在网格中依次点击若干方块，再点击“Advance”按钮若干次，单步运行程序，观察网格的变化。也可以点击 Start/Stop 按钮自动多步运行程序来看看网格是如何动作的。是什么原因让方块显示或者消失呢？

这个游戏有趣的地方在于它所使用的规则。上面这段代码的第 2 和第 3 行定义了游戏规则：

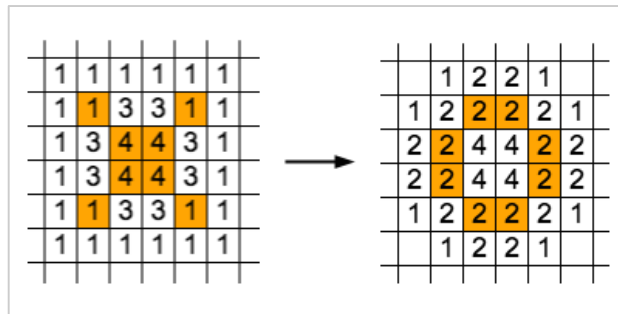
表 2-1 元胞自动机的游戏规则

代码第 2 行 survival = [2, 3]	如果某个方块是“活”的（橙色）状态，并且它相邻的方块中有 2~3 个是“活”的，那方块就保持“活”的或“幸存”状态。如这两种状态的方块：2、3，在程序运行后仍然是“存活”的状态。
代码第 3 行 birth = [3, 6]	如果某个方块是“死”（白色）的，但是若它相邻的方块中有 3 个或 6 个方块是“活”的，那么这个死的方块就会被激活。如这两种状态的方块：3、6，在程序运行后，这两种方块都会被激活。

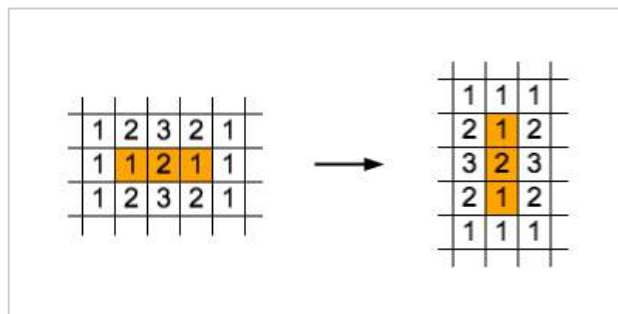
除了以上四种情况，其他状态的方块在程序运行后都会是“死”的状态（方块颜色从橙色变为白色）。这只是个仿真游戏，没有任何方块会被伤害哦！下面是一些例子：



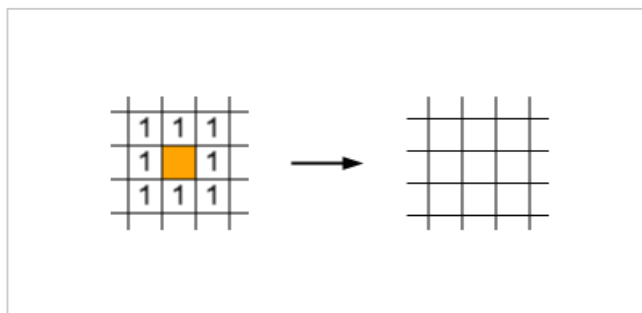
程序单步运行后，左图中“4”周围所有“3”的方块或者仍然保持“存活”状态或者被激活，“4”的方块会从“存活”的状态转为“死亡”状态，且由于它周围所有的方块都是“存活”状态，所以方块的数字从“4”变为了“8”，其余各方块的数字也随之发生了变化，最后结果如右图。（大家想想，如果再点击一次 **Advance**，方块又会有何变化？）



程序单步运行后，左图中所有“3”的方块全部被激活，其余处于“存活”状态的“1”和“4”的方块都会从“存活”状态转为“死亡”状态，各方块相应的数字也随之发生变化，运行结果如右图。（大家想想，如果再点击一次 **Advance**，方块又会有何变化？）



程序单步运行后，左图中处于“存活”状态的“2”仍然保持存活，处于“死亡”状态的“3”全部被激活，处于“存活”状态的“1”转为“死亡状态”。各方块上的相应数字也发生了变化，最后运行结果如右图。（大家想想，如果再点击一次 **Advance**，方块又会有何变化？）



上图中只有中间一个方块是“存活状态”，所有的方块都不满足“存活”或“复活”的条件，因此程序单步运行后，所有方块都是“死亡”状态，方块消失。

单击网格来创建自己喜欢的图案，你可以不断点击 **Advance** 来看看每个周期应用规则的结果，也可以点击 **Start** 来看看多个周期自动应用规则的结果。如此有趣的交互来自看似随意的规则。人们研究这些图案多年，并发现了一些特定的组合，就像在野外发现某种类型的动物一样。在网格上创建图案，点击 **Start** 来试着让图案应用规则后能满足下面这些类型：

- 稳定的-图案不随时间发生变化
- 振荡器-图案会随时间变化，但会在 **N** 个周期后重复
- 飞行物-图案本身不会发生变化，但是会像鸟和飞船一样移动位置
- 找出你自己的图案属于哪种类型

修改系统的另外一种方式是改变规则本身。上面仿真的例子中所应用的规则只是元胞自动机众多规则中为人所熟知中的一种，每一个新规则都会产生全新的属性集。上述的仿真使用的是一连串“S/B”规则，S 代表 **survival** 幸存者，B 代表 **births** 新生者。在这个“人生游戏”（**Game of Life**）的仿真中，S/B 的值是 23/36，意思是某个存活方块如果有 2 个或 3 个相邻的存活方块，该方块仍然是存活的，某个死亡的方块如果有 3 个或 6 个相邻的存活方块，该方块就会复活。你可以看看这些人们研究了多年的各种规则，并试着在仿真中修改第 2 和第 3 行代码来应用这些规则。如果你修改了规则，记住要点击 **Play** 按钮来重新运行代码。

数学是一个跨领域的课题，关于系统及其行为的研究已经影响并开拓了许多科学和人文研究领域。而很多新知的发现和新研究领域开辟，往往是由于新算法的设计和应用所产生的。

如果你对于元胞自动机以及生命游戏的研究感兴趣，你可以通过网络搜索这两个关键词来获得更多的信息。**Stephen Wolfram** 曾研发过一个为元胞自动机创建规则的系统，并对其性质进行了广泛的研究。你可以在 **Wolfram** 的 **Mathworld** 以及 [A New Kind of Science](#) 中找到该研究的更多的内容。当你在对学生进行教学的时候，可以使用该标准来评测本活动。

2-1-4 科学：基因组学

根据[人类基因组计划](#)的研究结果，人类细胞中 DNA 的碱基对长度大约在 5 千万到 2.5 亿之间。研究者们仍在不断加深对基因功能和基因表达方面的理解。编码在 DNA 中的信息量巨大，同时基因之间的相互作用也非常复杂。那么科学家是如何从如此多的可能性和数据中获得有用的结论的呢？算法的应用起到了很大的作用。即使在这些技术能被高效的使用之前，人们就已经尝试将算法应用在基因的研究中了。

格雷戈尔·孟德尔是研究豌豆杂交育种和古老选择育种技术的先驱。孟德尔所进行的试验中影响最为深远的部分，是他从结果中发现的模式。孟德尔研究了数千种植物，从中寻找那些特定性状被选择出的结果。通过研究这些数据，孟德尔发现了遗传的模式，并揭示了如今被称为遗传定律的规律。尽管旁氏表要在数年后才被发明出来，但是它能够帮助你将这些遗传可能性可视化。你可以想象这对孟德尔是一个多么大的挑战！

首先我们想象有一个可以决定花朵颜色的基因。如同人类一样，花从它的父母那里分别获得一个基因。这些基因可能彼此不同，使得花朵呈现出遗传的多样性。对于该颜色基因，我们假设它有两种表现型：紫花（P）和白花或者无颜色花（p）。当有机体中的两种基因都是白花基因（pp）时，花朵颜色呈白色。只要有一个颜色基因为紫色基因（Pp 或者 PP）时，花朵呈紫色。

下面的旁氏表展示了当父母分别具有两种基因时，子代的全部四种可能的基因型。这样的父母被称为杂合子。杂合子的子代有 75% 的可能性是紫色花。当父母中有某一方为纯合隐形或者纯合显性基因时，这一结果就会发生变化。当存在复数的基因时，其子代的基因型的可能性的数目也呈指数方式增长。一次双因子杂种杂交（两种基因）将会有 16 种可能性，而三因子杂交则有 64 种可能性。

表 2-2 旁氏表

	P	p
P	PP 紫色	Pp 紫色
p	Pp 紫色	pp 白色

你可以看到这些数据的采集量和计算量惊人的增长速度。很难想象为了采集七种基因类型结果，格雷戈尔·孟德尔在七年的试验中需要投入多大的精力和耐心。而当你想到人类的基因组包含有超过 20000 种基因时，你会发现纸、笔以及传统的计算方式已经无法满足基因研究的需求。然而，当你关注这些研究数据时，你会跟孟德尔一样发现：数据中的模式可以用来归纳法则，并帮助我们理解其中的现象。

算法是一系列具体指令的集合，这些指令可以通过对数据进行模式搜索、计算概率以及其他方式来帮助我们理解这些数据。甚至人类细胞在进行 DNA 复制和蛋白质合成时使用的过程，也是某种转录和翻译算法。该过程的每一步都可以被清晰的记录和理解。人类的每一个细胞每天无时无刻不在重复着这一过程，而这清晰的向我们展示了算法的威力。理解这个过程能够帮助科学家们更好的了解人体的生理过程，并能够帮助我们找到对抗病毒的新方法。

转写和翻译的实例

以下是一个用代码模拟 DNA 转写成信使 RNA，再将信使 RNA 翻译成氨基酸链的例子。我们的细胞每天都在经历这一过程来合成身体所需的蛋白质。通过模拟这一过程，我们将能够观察到单个突变对氨基酸链的影响。

在以下的实例中，第一个 DNA 片段来自血红蛋白的 DNA 编码（HBB）。第二个和第三个 DNA 片段大体上是相同的，除了第二个片段中的第五个碱基对由“C”被改为“T”而第三个片段的“C”被直接删除。

转写和翻译算法的实例：

```
#DNA --> mRNA
```

```
transcription_dict = {'A': 'U', 'G': 'C', 'C': 'G', 'T': 'A'}
```

```
#mRNA --> Amino acid
```

```
translation_dict = {'UUU': 'Phe', 'UUC': 'Phe', 'UUA': 'Leu', 'UUG': 'Leu', 'UCU':  
'Ser', 'UCC': 'Ser', 'UCA': 'Ser', 'UCG': 'Ser', 'UAA': 'Stop', 'UAG': 'Stop', 'UAU':  
'Tyr', 'UAC': 'Tyr', 'AUU': 'Ter', 'AUC': 'Ter', 'UGU': 'Cys', 'UGC': 'Cys', 'ACU': 'Ter',  
'UGA': 'Stop', 'UGG': 'Trp', 'CUU': 'Leu', 'CUC': 'Leu', 'CUA': 'Leu', 'CUG': 'Leu',  
'CCU': 'Pro', 'CCC': 'Pro', 'CCA': 'Pro', 'CCG': 'Pro', 'CAU': 'His', 'CAC': 'His',  
'GUU': 'Gln', 'GUC': 'Gln', 'CGU': 'Arg', 'CGC': 'Arg', 'CGA': 'Arg', 'CGG': 'Arg',  
'AUU': 'Ile', 'AUC': 'Ile', 'AUA': 'Ile', 'AUG': 'Met', 'ACU': 'Thr', 'ACC': 'Thr', 'ACA':  
'Thr', 'ACG': 'Thr', 'AAU': 'Asn', 'AAC': 'Asn', 'AAA': 'Lys', 'AAG': 'Lys', 'AGU': 'Ser',
```

```
'AGC': 'Ser', 'AGA': 'Arg', 'AGG': 'Arg', 'GUU': 'Val', 'GUC': 'Val', 'GUA': 'Val',
'GUG': 'Val', 'GCU': 'Ala', 'GCC': 'Ala', 'GCA': 'Ala', 'GCG': 'Ala', 'GAU': 'Asp',
'GAC': 'Asp', 'GAA': 'Glu', 'GAG': 'Glu', 'GGU': 'Gly', 'GGC': 'Gly', 'GGA':
'Gly', 'GGG': 'Gly'}
```

```
def transcription(DNA):
    #Return a string of RNA given a string of DNA
    mRNA = ""
    for base in DNA:
        mRNA += transcription_dictionary[base]
    return mRNA

def translation(mRNA):
    # Return a list of amino acids given a string of mRNA
    amino_acid_chain = [ ]
    codon_length = 3
    end = len(mRNA)
    ribosome_one = 0
    ribosome_two = 3
    while ribosome_two <= end:
        codon = mRNA[ribosome_one: ribosome_two]
        if codon in translation_dict and translation_dict[codon] is not 'Stop':
            amino_acid = translation_dictionary[codon]
            amino_acid_chain.append(amino_acid)
            ribosome_one = ribosome_two
            ribosome_two += codon_length
    return amino_acid_chain
```

结果:

血红蛋白 DNA 片段

DNA #1 : GGATCCTCACATGAGTTCAGTATATAATTGTAACAGAATAAAAAAT

mRNA: CCUAGGAGUGUACUCAAGUCAUAUAUUAACAUUGUCUUAUUUUUUA

Amino Acid Chain: Pro-Arg-Ser-Val-Leu-Lys-Ser-Tyr-Ile-Asn-Ile-Val-Leu-Phe-Phe

第五碱基对从'C' 变更为 'T'后的血红蛋白 DNA 片段

DNA #2 : GGATTCTCACATGAGTTCAGTATATAATTGTAACAGAATAAAAAAT

mRNA: CCUAAGAGUGUACUCAAGUCAUAUAUUAACAUUGUCUUAUUUUUUA

Amino Acid Chain: Pro-Lys-Ser-Val-Leu-Lys-Ser-Tyr-Ile-Asn-Ile-Val-Leu-Phe-Phe

第五碱基对被删除的血红蛋白片段

DNA #3 : GGATCTCACATGAGTTCAGTATATAATTGTAACAGAATAAAAAATC
mRNA: CCUAGAGUGUACUCAAGUCAUAUAUUAACAUGUCUUAUUUUUUAG
Amino Acid Chain: Pro-Arg-Val-Tyr-Ser-Ser-His-Ile-Leu-Thr-Leu-Ser-Tyr-Phe-Leu

这几种变异，甚至只是很微小的一个碱基对的变化或删除，都能对最终的蛋白质产物造成显著的影响。因此，如果一个遗传学家想通过手动过程来对这些进行研究的话，工作量是十分巨大的。与之相比，使用算法和代码来模拟这一过程就能让我们及时的看到这些变异对蛋白质产物的影响。将算法整合进科学领域能够帮助科学家更有效率的进行研究，并节省下大量的实验和分析时间。目前为止，人们还在对应用算法的成本和可行性进行优化。当你和你的学生们能够探索这些基因数据时，你们不仅可以了解到基因学的知识，还可以进行属于你们自己的研究，甚至有所发现。

如果你们对于如何将算法应用到基因研究的领域感兴趣，[Rosalind Problems](#) 会是一个很好的切入点。最开始的几个 **Rosalind problems** 是生物学课程的公共部分。学生不仅仅能从中学习到相关知识，而且可以学习到如何设计算法来解决这一类问题。即使你们没有真正使用 **Python** 或者其他编程语言来实现这些算法，你的学生们也可以通过写下解决问题的步骤，来加深对这些过程的了解。

你可以从 [BLAST](#) 和 [Google Genomics API](#)（或者参考附件六）中找到那些被应用于生物信息学的工具。你也可以从 [Fold.it](#) 中亲身体会到算法的威力，甚至能够参与到生物学研究中来。这里有一些搜索关键词能为你提供更多的信息：旁氏表（**Punnett Squares**），生物信息学（**Bioinformatics**），计算生物学（**Computational Biology**），和人类基因组计划（**Human Genome Project**）。当你在对学生进行教学的时候，可以使用该标准来评测本活动。

2-2 课程反思

定义一种算法的方法之一就是通过创建一连串的逐步指令，并使用这些指示去解决一个问题或者执行某个任务。这一章节的目的就是展示算法的可能性。你也许在之前曾经体会过某个算法的威力，比如当你使用搜索引擎很快找到你想要的信息时，或者你使用的 **GPS** 设备基于交通状况给出一个更好的路线时。你也许曾经意识到，开发一个算法或者对算法进行测试是不需要具体的代码的。然而，使用编程技术则可以比手动更快更好的执行算法的许多步骤。

旅行问题

旅行推销员问题是计算机科学家所提出的一个经典挑战：为一名旅行推销员规划出一条穿过一

个国家的最优路线。对于该问题来说，找到一条可行路线并不难。你只需要按照你自己的步骤创建一条经过所有城市的路线即可。然而，该问题的困难之处在与：如何确定一条最短路线或者步骤最少的路线，以及是否能在合理的时间内求得一个最优可行方案。

与时俱进的字词

当我们将算法用计算机技术实现的时候，我们会发现指令的执行要比人工执行快得多。这样的效率优势在处理大规模数据的时候尤为明显。以往需要大量人手共同工作数年才能得出的结果，如今在很短的时间就能得到。一个经典的计算机应用案例就是我们身边的计算器：从前需要许多人共同合作才能完成计算量，在今天可以用任何袖珍计算器迅速完成。[N-gram Viewer](#) 就是一个将算法应用到大规模数据集合中，以获得某些文化中有趣甚至是滑稽的发现。随着越来越多的人了解如何分析数据和整理信息，他们将能够把这些技能应用到新的领域当中，并帮助他们以一种以往难以想象的方式拓展对于宇宙的理解。

元胞自动机

给定一个系统一些初始参数，并规定系统的运行规则，然后让该系统自行运作——该过程我们称之为混沌系统。对于元胞自动机来说，任何一点初始细胞生存状态的或细胞繁殖的规则的变化，都会导致结果的巨大变化。混沌系统广泛存在于天气、交通模式、宇宙学和基因科学中。

基因组学

农民们早已知道动物和植物们已经用相似的特性繁衍后代长达一千多年了。当人们发现所有的生命都使用相同的基因编码和遗传信息时，他们意识到其中蕴含着探索生命奥秘的巨大机会。从苍蝇的研究中获得的知识可以被应用到猪和其他的物种当中。更加深入的研究人类基因组，了解人类基因组是如何随着时间而变化的，以及发现不同文化的人类种群间的基因差异，都有助于我们改善医疗技术和为人类这个种族留下历史记录。人类的 **DNA** 包括数千种不同的基因，而每一种基因都可能需要耗尽一个人的一生来进行研究。算法已经被应用到快速搜索和分析生物样本之中，这能帮助科学家寻找模式与基因突变和某些治疗手段造成的影响。没有任何两个人有着完全一样的基因表现型。深入对于基因的研究将有助于人们了解自身的独特性，并且体会到个体是如何融入更广大的生命群体中的。

三、模式发掘

3-1 概述

在本节中，你可以试着参与到一些能被应用到你的课堂中的活动中来。这些活动涉及到编辑已有的代码，但并不需要你事先对编程有所了解。这是一个体验不同领域中计算思维过程的好机会，所以请不要有顾虑的去尝试那些你不熟悉领域中的活动吧。点击一个活动以开始课程，探索并享受其中的乐趣吧！



图 3-1 数据压缩案例示意图

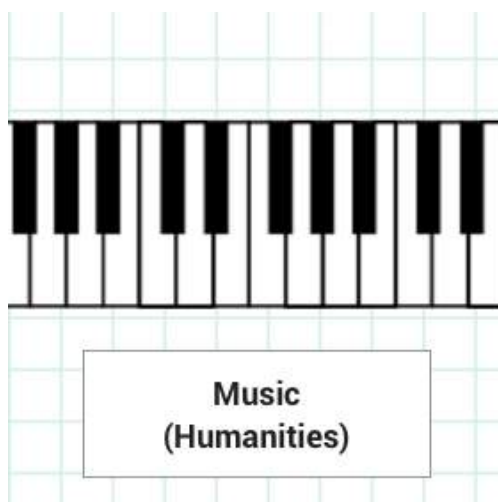


图 3-2 音乐案例示意图

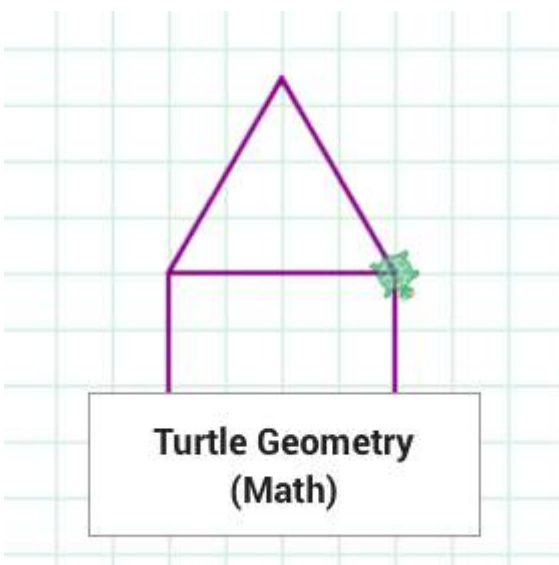


图 3-3 小海龟几何案例示意图



图 3-4 分类案例示意图

3-1-1 计算机科学：数据压缩

就在不久之前，内存容量和存储空间大小仍然是困扰手机和电脑消费者的主要因素。现如今，当你在考虑购买相关设备，尤其是准备购买移动设备并准备在里面存储大量多媒体信息时，存储空间的大小依旧是一个需要考虑的重要因素。在当今时代，有很多的信息和数据都来源于网络。开发者们不得不思考如何才能压缩数据规模，以避免存储空间消耗殆尽和支付昂贵的存储成本。

信息可以通过多种途径被压缩。对于文本信息，你可以使用缩略词来进行压缩，诸如 DNA、3D、TTYL、assoc、orig、gov 等。搜索阿雷西博信息（Arecibo message），你可以看到一个尝试用很小的空间就可以传输人类文化和信息中关键部分的例子。你也许听过这么一句话：一张照片胜过千言万语。图像确实能够传输大量关于复杂的场景或情感内容的信息。你会在网页中、社交网络中、短信中使用图片。而在你使用图片的时候，通常需要经过一个上传和下载信息的过程。根据图像或视频的质量和尺寸的不同，这个信息传输过程可能会花费你大量的时间和金钱。当软件开发涉及到使用多媒体信息时（例如文本、图片、视频），开发者们通常都要考虑应该提供什么品质的信息给接收者，以使它们愿意下载或使用流媒体播放。

图像在屏幕上是以像素集合的形式显示出来的。图片的像素就是数据的比特位，为了降低存储图片所需的空间，人们可能会使用一种叫做位掩码的技术。在图 3-5 中，你应该可以看到两幅相似的图片，其中标识有两幅照片分别占据的字节数，也就是存储空间大小。事实上，这两幅图片完全相同，因为表示图片的比特数并没有任何的减少。

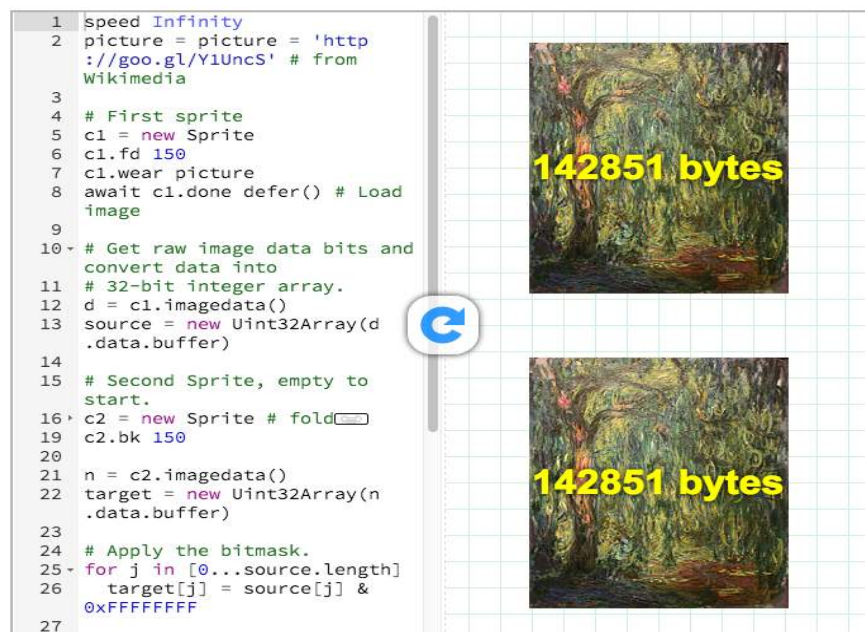




图 3-5 数据压缩案例代码与效果示意图

互动案例: <https://computationalthinkingcourse.withgoogle.com/unit?lesson=13&unit=12>

互动链接: <https://jisuansiwei.pencilcode.net/edit/CSDataCompressionActivity>

位掩码在图片像素的比特位中运用了某种逻辑运算。这个逻辑运算将原始图片的比特位与位掩码按位进行“&（与）”操作。只有当源文件中的比特位和位掩码对应的比特位均为 1 时，目标文件的对应比特值才是 1。上述代码中第 26 行（`target[j] = source[j] & 0xFFFFFFFF`），试着将 `0xFFFFFFFF` 改为 `0xFF000000`，点击 play 来重新运行代码来看看结果如何。

```
1 speed Infinity
2 picture = picture = 'http
  ://goo.gl/Y1UncS' # from
  Wikimedia
3
4 # First sprite
5 c1 = new Sprite
6 c1.fd 150
7 c1.wear picture
8 await c1.done defer() # Load
  image
9
10 # Get raw image data bits and
   convert data into
11 # 32-bit integer array.
12 d = c1.imagedata()
13 source = new Uint32Array(d
  .data.buffer)
14
15 # Second Sprite, empty to
   start.
16 c2 = new Sprite # fold
19 c2.bk 150
20
21 n = c2.imagedata()
22 target = new Uint32Array(n
  .data.buffer)
23
24 # Apply the bitmask.
25 for j in [0...source.length]
26   target[j] = source[j] &
     0xFF000000
27
```



-166, -29

注意：第一组 FF 表示的是像素不透明度的比特值，FF 表示图片完全不透明，这样我们就可以看清楚生成的图像（如果值是 00，生成的图像就是全透明的）。前缀 0x 表示这一串数字是 16 进制的。

背景知识

将像素点表示为二进制和十六进制的数。

一张图片在屏幕上是通过一组像素在屏幕上显示出来的。每个像素可以用很多方法表示。每个像素包含了颜色的 RGB（红、绿、蓝）值及不透明度值（图片的透明度）的信息。以下是表示一个像素颜色信息的几种方法：

- 使用二进制数值来表示颜色信息，每个二进制位（比特位）由 0 或 1 来表示。

红	1111 1111 0000 0000 0000 0000
绿	0000 0000 1111 1111 0000 0000
蓝	0000 0000 0000 0000 1111 1111

- 使用十六进制数值来表示颜色信息，十进制中的数字 0-15 分别由 0-F（1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F）来表示。

红	FF0000
绿	00FF00
蓝	0000FF

通过改变代表颜色信息的那些数值，你可以让显示屏显示出来任何你熟悉的颜色。当你试图改变一个网页的颜色的时候，你可以改变代表背景颜色的颜色数值或者代表文字颜色的数值。如果你想将一个网页的背景颜色设置为白色，你很有可能会使用十六进制数值#FFFFFF，其将代表背景像素的红，绿，蓝三个颜色的数值设置成 100%，这意味着屏幕上显示为白色。反过来，如果你想将网页上的文字设为黑色，你可以使用#000000，其将代表文字像素的红，绿，蓝三个颜色的数值设置成 0%。

如果你对于如何将一个数字通过二进制或者十六进制表示出来，请查阅 Wikipedia 网页上关于十六进制的部分和 Math is Fun 相关的网页。

位掩码为 0xFF000000 时，红绿蓝三色的位掩码比特位的值全部为 0。位掩码运算结束后，正如下表中所看到的一样，该像素的每一个 bit 位的值都变为 0，没有留下任何的顏色信息。这是因为在二进制的&（与）操作中，只有当两个操作数均为 1 时，结果才是 1。因此，0&0=0，

0&1=0，1&1=1。由此生成的图像基本是空白的，只保留了少量字节以表明这是一张图片。如下图：

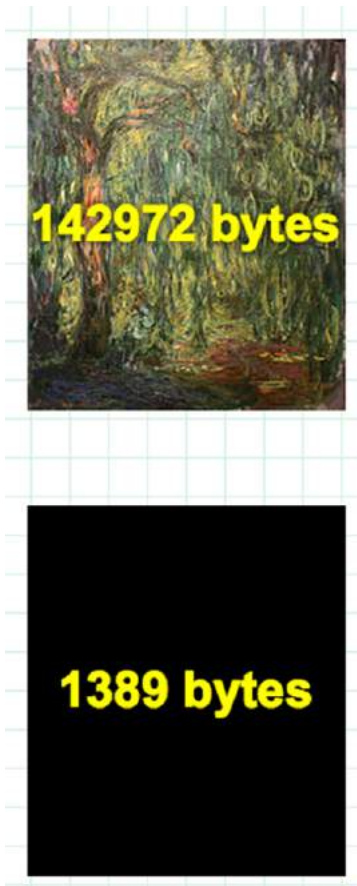


表 3-1 位掩码是 0xFF000000 的运算结果

Red 红	1111 1111 0000 0000 0000 0000
& Bitmask 位掩码	0000 0000 0000 0000 0000 0000
Result 结果	0000 0000 0000 0000 0000 0000

如果你将位掩码改回一开始的 0xFFFFFFFF，运算后将不会改变像素的原始信息，而这就是为什么图片的尺寸跟原本的一样。这是你在该活动一开始所看到的。

表 3-2 位掩码是 0xFFFFFFFF 的运算结果

Red 红	1111 1111 0000 0000 0000 0000
& Bitmask 位掩码	1111 1111 1111 1111 1111 1111

Result 结果	1111 1111 0000 0000 0000 0000
-----------	-------------------------------

现在，你已经看过了两个极端的例子。当使用位掩码 `0xFF000000` 的时候，没有任何比特位能够通过该位掩码，这使得图像会最大程度上减少存储所需的存储空间，但这是以牺牲所有的信息和品质为代价的。当使用位掩码 `0xFFFFFFFF` 的时候，所有的比特位都可以通过而所需的存储空间没有任何变化。

通过改变位掩码的值，你可以自己尝试一些其他组合。通过不同的尝试，你可以发现其中的一些模式，使得你可以在图片质量和存储空间做出最佳的取舍。如果你需要关于二进制和十六进制互相转换的帮助，你可以搜索二进制转换成十六进制，或者十六进制转换成二进制。

现在你已经尝试过不同的组合了，你在其中找到了什么模式了吗？你有没有找到一个位掩码，使得既能保证图片有足够的质量同时也能大幅度减少所需的存储空间？这里有一些组合你也许可以试一试。记录下所需的存储空间和图片质量（尺度为 1-10）。在你改变了比特位的值以后，按 **play** 来再次运行代码。

表 3-3 不同位掩码时存储值与品质实验记录

位掩码的值 (base ₁₆)	存储值 (bytes)	品质 (1-10)
0xFFFFFFFF	142845 (original size)	10
0xFF000000	1389	1
0xFFAAAAAA		
0xFFCCCCCC		
0xFFEEEEEE		
0xFFA0A0A0		
0xFFC0C0C0		
0xFFF0F0F0		

哪一个位掩码的值你认为是最优的？或者你有没有发现其它你觉得更好的位掩码值？

也许对于不同的使用者来说，图片的品质是一个非常主观的概念。你在尝试不同位掩码的时候发现了什么模式了吗？试着将其归纳概括出来，好让你能够向其他人解释如何在维持一定图片质量

的同时，尽可能的减少所需要存储的信息。

如果你正在开发一款用来传输图片的软件，而不是开发一款图像编辑软件，你可能对图片质量有着不同的需求。改变代码第二行中的图片 **URL** 地址以指向其它类型的图片，你可以看到这一技术是否能被应用到高分辨率的图像和黑白位图中。理解这些一般原理能让你更好的开发一个算法，并调整它以适应你的需求。

在本活动中，你在存储空间和图片质量中寻找应用位掩码的模式时使用了计算思维。这些模式帮助你确定你想使用的位掩码值，并能够帮助你创建一些普遍规则用来应用位掩码以减少图片的存储空间。如果你对该课题感兴趣，你可以搜索如下关键词来获得更进一步的信息：阿雷西博信息、位屏蔽、位图、色彩量化、多色调分色法、递色、媒体计算。

不插电的计算机科学（**CS Unplugged**）也有一个关于[文本内容压缩的活动](#)。当你在对学生进行教学的时候，可以使用该标准（或者参考附件二）来评测本活动。

3-1-2 人文学科：音乐

喜爱音乐是人类的天性。在所有的文化中，各个年龄层的人都在制作和聆听音乐，无论这些音乐是用手有节奏的敲击桌子还是用各种乐器精心演奏的交响乐。

音乐是各个文化的产物。那么我们是否能在大多数音乐中找到一些共有的模式或者一般原理呢？通过应用计算思维，你可以创作一首歌曲并使用发现的模式和原理来改良它。请带上你的耳机，打开你的麦克风。然后按下图中的“**Play**”按钮来创作一些音乐。

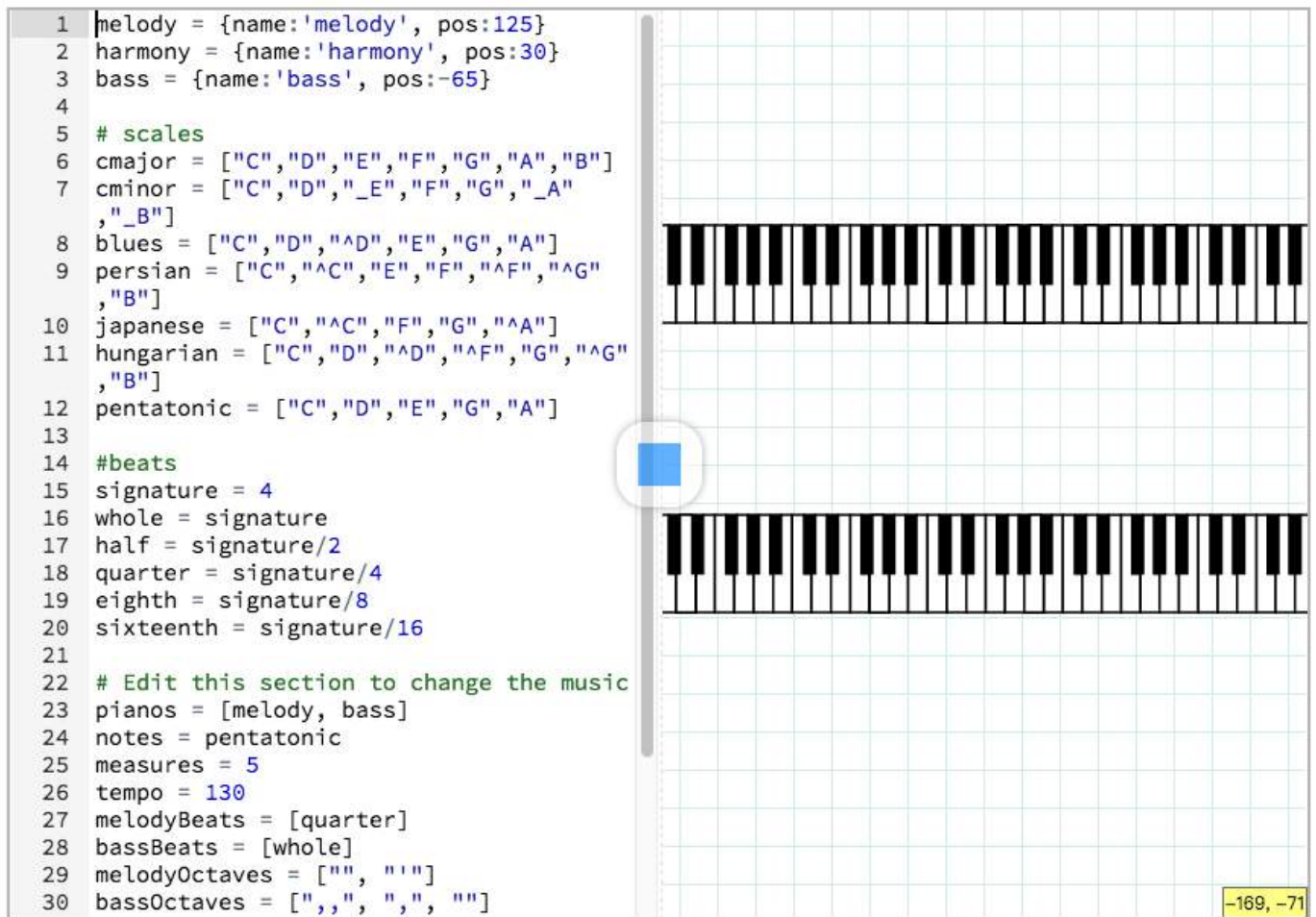


图 3-6 音乐案例代码与效果示意图

互动链接: <https://jisuansiwei.pencilcode.net/edit/HumanitiesMusicActivity>

以上的代码会从一个五声音阶中随机的选取一些音符。五声音阶包括"C", "D", "E", "G", "A" 五个音。世界上的许多文化都使用五声音阶来创作音乐。音阶的设置代码的第 24 行, 你可以通过更改这里的内容来使用任何你想使用的其它音阶。例如, 删除 `notes = pentatonic` 并改写为 `notes = japanese`。

每当你改变了代码的内容的时候, 请确认都按下了“play”来运行代码。因为代码只是随机挑选的音符, 所以所生成的音乐并不像一个优秀的作曲家所创作的乐曲那样。然而, 这里有些方法可以让我们生成的音乐更加动听。

节拍

高低长短不同的音符组成了音乐的旋律。当你以每秒四拍的节奏唱一首歌, 你会发现这跟你以每秒一拍的节奏唱一首歌完全不同。音乐家们使用共同的语言来创作和讨论音符的节拍, 所以他们能够互相理解别人的作品。其中的一些拍子为全音符、二分音符、四分音符、以及八分音符, 你可

以把它们理解为每秒一拍、每秒两拍、每秒四拍、每秒八拍。我们现在的仿真程序使用了四分音符作为音乐旋律的主要节奏，并在低音部使用了全音符用作音乐的节奏。

举例来说，你可以通过改变代码 27 行中的 `melodyBeats = [quarter, eighth]`，在旋律中添加八分音符，或者用同样的方法在 28 行中改变 `bassBeats`。

音程和和弦

你也许会发现单个音符听起来还不错，但是当某些音符被一起演奏的时候，音乐的感觉就会被改变。两个音符同时被演奏或依次演奏被称为音程。而当三个或更多的音符被同时演奏的时候，我们称之为和弦，这其中有许多种可行的配置。在上面的仿真中，每一个音阶中的音符都有同样的机率被选中。试着在代码第 31 行中改变 `biasForNoteson` 的值。其中的每一个数字都对应着一个音阶中的音符每次被选中加入乐曲中的概率。这些值都小于 1，且所有的值相加后应为 1。

这里有一个概率配置 `biasForNotes = [0.5, 0.0, 0.0, 0.5, 0.0, 0.0, 0.0]`。在该配置中，仅有第一个音符和第四个音符会被选中加入音乐中。试着用不同的概率配置并找出你最喜欢的一个。在你调整完代码后，按“Play”以运行代码。

和声

两个是否一定强于一个？许多乐队都有不止一个歌手，而且在一个管弦乐队，很多的乐器会同时演奏相同或相似的音符。这种演奏方法被称为和声。和声可以在合唱中添加丰富的声音，或者当音符为互补的时候，在声音中添加新奇的复杂的效果。

通过在代码第 23 行中添加 `pianos = [melody, harmony, bass]`，可以在代码中添加另一个钢琴来在你的音乐中引入和声。在你调整过代码之后，按下“play”来重新运行代码。在添加和声后，你是否还想改变诸如偏好或拍子等其它变量？

除了上述提到的以外，你还有很多可行的调整或添加其它内容来使代码生成更加出色的音乐。你可以尽情按自己的意愿来“玩”这些代码。如果你遇到了什么麻烦，你可以通过刷新页面或者尝试再次运行更改后的代码来解决。在上面的活动中，你将一首歌曲解构成不同的成分，并尝试进行调整。在经过多次尝试以后，你也许会发现某些设置比其它设置能生成更好的音乐。

这些音乐制作的原理众所周知，音乐创作者可以通过算法生成一段旋律或“hook”来让歌曲更加吸引人。

译者注：hook 是一种音乐的表现形式。详情可百度“音乐 hook”。

如果你想了解更多关于音乐中的模式和定律的信息，你可以搜索如下关键词：五度循环, 毕达哥拉斯的调谐, 和弦, 算法作曲, 媒体计算。当你在对学生进行教学的时候，可以使用该标准（或者参考附件二）来评测本活动


3-1-3 数学：小海龟几何

在几个星期的培训以后，你已经成功让宠物小海龟学会根据你的指令咬上笔，前进以及转弯。现在，看看你是否能教会小海龟画出一些形状图案。

试着向小海龟发出指令，让它画出来一个正方形。使用 **fd** 指令来让小海龟前进一个指定的距离，使用 **rt** 指令来让小海龟转动一个指定的角度，如下例：

```
fd 100
```

```
rt 90
```

fd 100 这条指令是让小海龟在屏幕上前进 100 个像素，而 **rt 90** 让它右转 90 度。当你添加了更多的指令，点击运行按钮  运行指令，看看会发生什么。

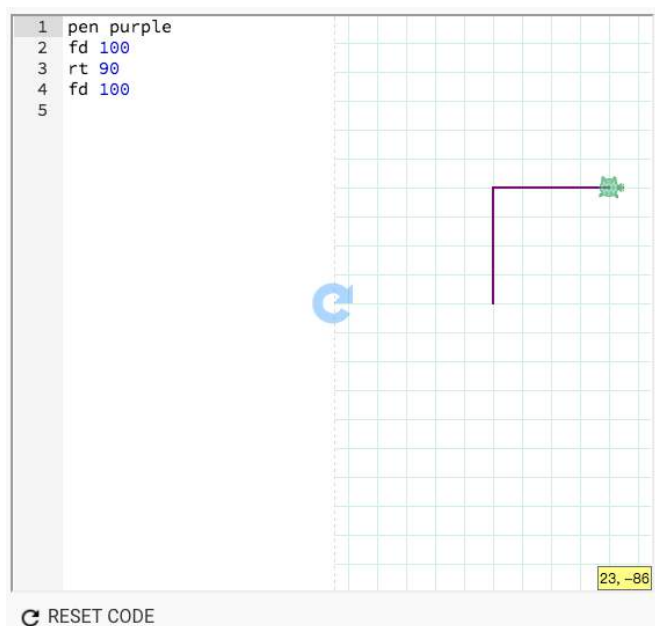


图 3-7-a 小海龟几何案例代码与效果示意图

互动链接：<https://jsuansiwei.pencilcode.net/edit/MathTurtleGeometry>

为什么不给小海龟一个可以睡觉的家呢？继续添加你的指令，看你是否可以让你的小海龟画出来一个如同下图一样的房子。如果你的小海龟第一次没有做出来，那没有什么，再来一次，继续努

力!

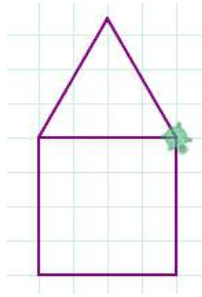


图 3-7-b

现在你的小海龟已经训练的很好了，只要你告诉它需要重复的步骤和次数，它就可以按部就班地执行了。使用命令 **for [a..b]** 来重写画房子的指令，告诉小海龟循环开始和结束的数值以确定循环的次数。你可以使用上面学到的这些指令来设计一个算法，画出任何你想要的常见的正多边形。

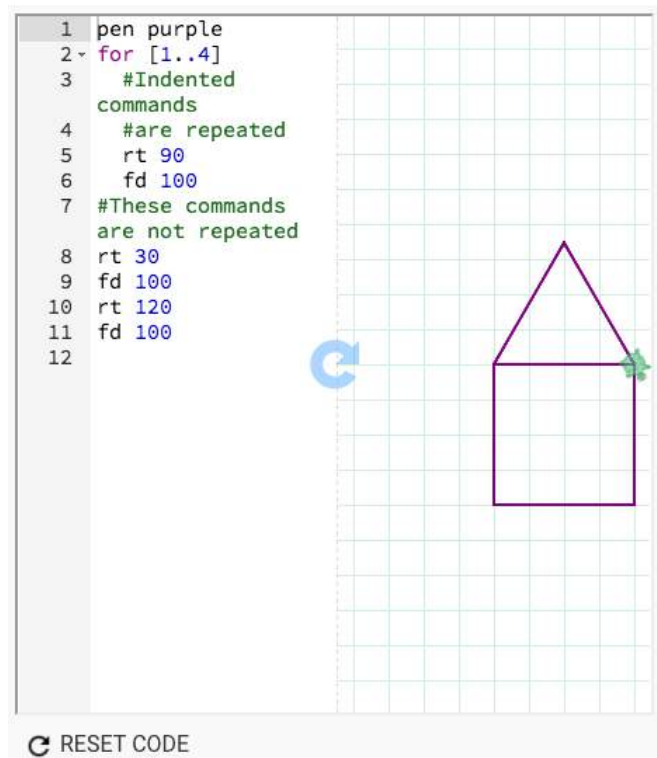


图 3-7-c

你可以把你的计算思维“灌输”给小海龟，让它能画出任意边数的正多边形（如正五边形，正十二边形等等）。为了让这个挑战更可控，将它分解成一些比较小而容易解决的简单任务是非常必要

的。将一个复杂任务分解成一系列简单小任务的过程就叫做**任务分解**。

下面是一个画一个正方形的指令：

```
Fd 100    #前进 100 步  
rt 90     # 右转 90 度  
fd 100    #前进 100 步  
rt 90     #右转 90 度  
Fd 100    #前进 100 步  
rt 90     #右转 90 度  
fd 100    #前进 100 步  
rt 90     #右转 90 度
```

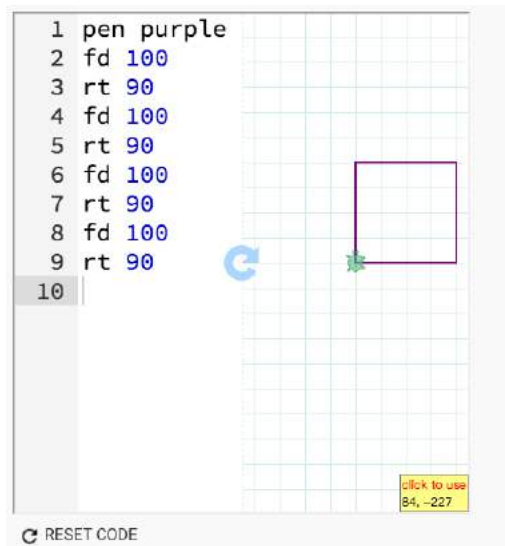


图 3-7-d

下面的步骤是完成这个任务的一个简单版本：

重复执行下面的指令 4 遍：

```
fd 100    #前进 100 步  
rt 90     #右转 90 度
```

在纸上写下画一个五边形所需要的代码。

如果你已经写出了一些代码，可以将你的想法在 **Pencil Code** 网站上用小海龟测试。做错了不要怕，这是好事，因为你可以从错误中找到正确的方法。

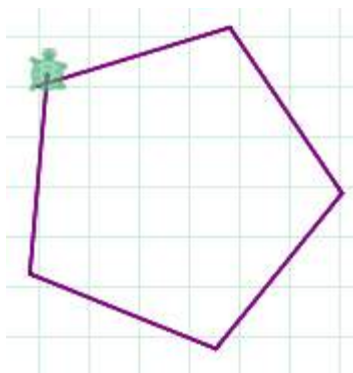


图 3-7-e

对比一下你设计的画五边形和正方形的步骤的异同。你是否有注意到多边形的边的数目和你步骤重复次数之间存在某种关联或者模式？

相等：代码行数和多边形边数相等（4 行代码对应一个四边形，5 行代码对应一个五边形）

两倍：代码行数大约是多边形边数的两倍（8 行代码对应一个四边形，10 行代码对应一个五边形）

三倍：代码行数大约是多边形边数的三倍（12 行代码对应一个四边形，15 行代码对应一个五边形）

这里是一个五边形指令的例子

```
1. fd 100  # 前进 100 步
2. rt 72    # 右转 72 度
3. fd 100  # 前进 100 步
4. rt 72    # 右转 72 度
5. fd 100  #前进 100 步
6. rt 72    # 右转 72 度
7. fd 100  # 前进 100 步
8. rt 72    # 右转 72 度
9. fd 100  # 前进 100 步
10. rt 72   #右转 72 度
```

四边形使用了 8 行代码，五边形使用了 10 行代码。点击下一步返回，描述一下边的个数和代码行数的关系。

两倍：代码行数大约是多边形边数的两倍（8 行代码对应一个四边形，10 行代码对应一个五边形）

对数据进行模式识别是计算思维另一个关键步骤，因为它提供了能揭示潜在原理、规律的关键信息。现在你已经知道步骤数量和角度之间存在关系了，这样你就可以借助这个**一般原理建立算法**，解决类似的问题。

只需要进行两处修改，上面给小海龟建房子的代码就可以实现代码重用。这可以推广到任意正多边形的情况。

在图 3-7-g 的代码中，改变 `sides=4`。接下来，重写 `for` 语句，将边数从 4 改为变量“`sides`”，这样命令就会重复和边数同样多的次数。

你也许已经发现转动的角度是 $360 / \text{边数}$ ，这是因为你的小海龟要转一圈回到起点。所以把第三行从 `rt 90` 变到 `360/sides`，就可以根据边的数量计算出角度的数值。

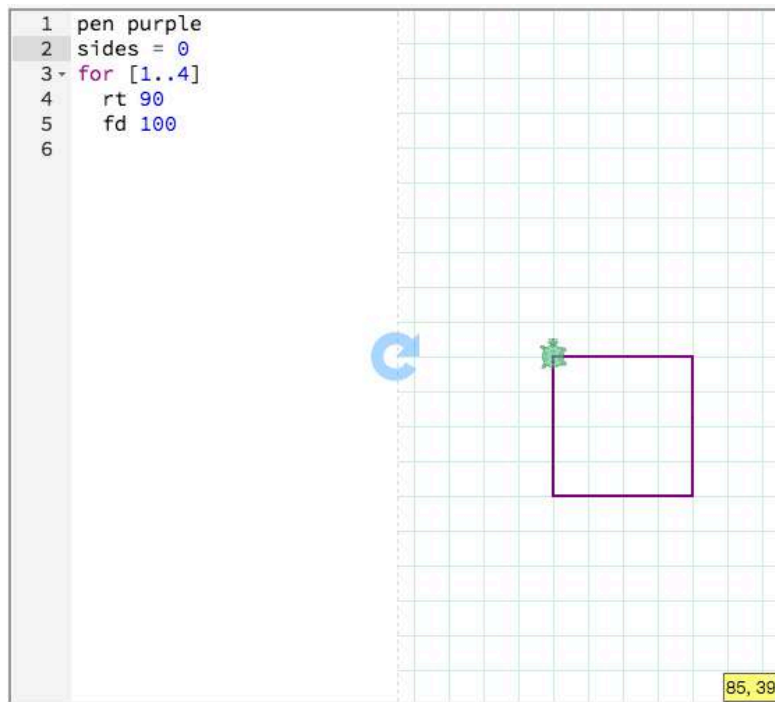


图 3-7-f

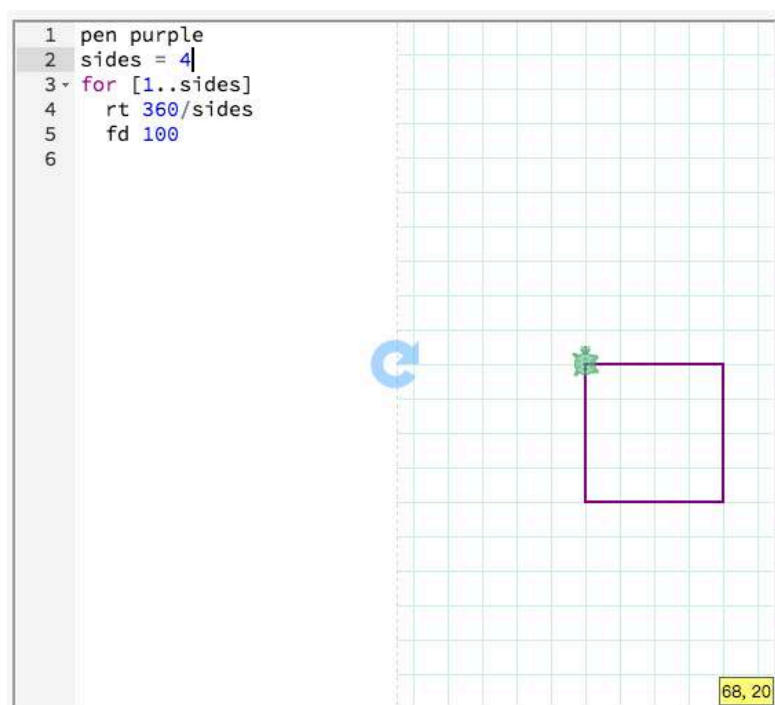


图 3-7-g

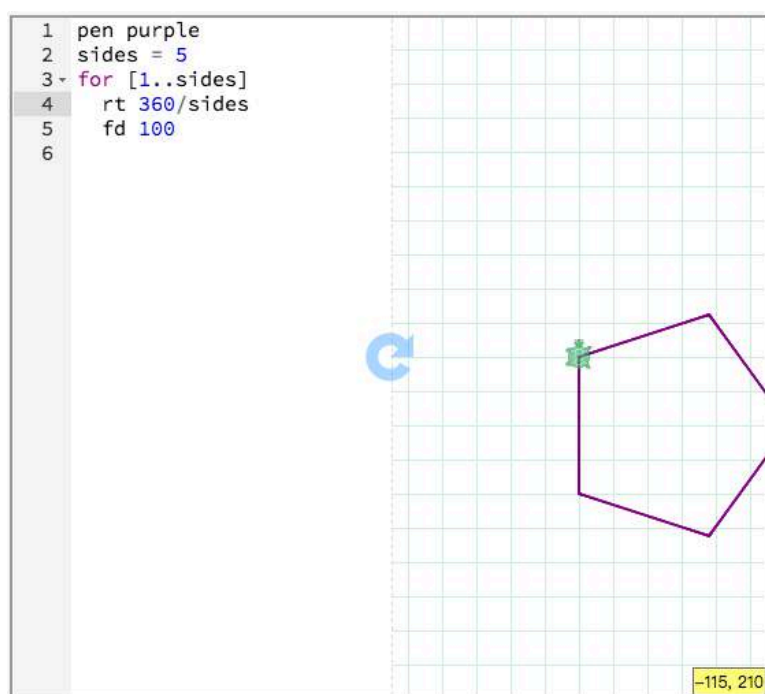


图 3-7-h

通过命令控制小海龟画图是经典的算法学习方式。学生需要设计画图的算法，才能创造出自己独特的作品。事实上，**Logo** 语言或小海龟有各种不同编程语言的版本。根据偏好或者需求，你可以继续探索使用纸笔代码（**Pencil Code**）或 **Blockly** 游戏进行编程教学。**CS unplugged** 中也有画图活动。如果你对用算法解决数学问题感兴趣的话，可以试试欧拉计划的问题（**Project Euler problems**）。你可以尝试使用 **if** 指令来标准化这些联系。

3-1-4 科学：分类



你有没有听过过一个游戏叫：**20 个问题**？在这个游戏中，一个玩家在心中想着某个东西，例如一架飞机，另一名玩家可以通过询问 **20** 个是非问题来猜测对方心中所想的東西。最近一项研究表明，地球上的不同种类的生命大约有八百万到九百万之间。

你认为需要多少个问题可以让你精确定位到任何一个心中所想的物种呢？



5 个吗？



这是一个巨大的挑战，但是我们可以用**分解法**解决这个问题，所谓分解法就是把一个大问题分解成小问题。我们可以先通过一个小例子来了解一下这种方法的原理。



图 3-8-a 树、鱼、熊、鸟、花、海豚、老虎、公鸡

如图 3-8-a，我在想以上其中一种生物，看看你能否通过只问 3 个问题来猜出我想的是什么？



问：这个生物有腿吗？



答：是的，我想的这种生物有腿。

于是剩下了：熊、鸟、老虎、公鸡，如图 3-8-b。



图 3-8-b 熊、鸟、老虎、公鸡



再问我另一个问题看看能否猜出我想的是哪种生物。你还可以问两个问题。



问：这种生物有皮毛吗？



答：是的。我想的这种生物有皮毛。

于是剩下了熊和老虎，如图 3-8-c。



图 3-8-c 熊和老虎



再问我另一个问题看看你能否猜出我想的是哪种生物，你还可以问一个问题。



问：这种生物的身上有条纹吗？



答：是的，我想的这种生物身上有条纹。

于是就剩下了老虎，如图 3-8-d。



图 3-8-d 老虎

只用了 3 个问题，你能从 8 个选项中猜出我在想的是老虎。

既然你已经看见了这个过程，你就可以运用这一概念，根据元素的属性指出元素周期表的元素，正如门捷列夫在发明元素周期表时所做的一样。

除了将这些元素按照原子质量排列外，（质子直到近 50 年后才被发现），你可以看到他还在表中留了很多空位给那些他预言到的但是还未被发现的元素。

计算思维不总是和电脑或是否运用了技术有关。它是一种几百年来一直为科学家们所使用的过程。在传统科学方法各个环节中，计算思维最适合在实验及分析阶段使用。

在前面提到的两个例子中，你接受了一项艰巨的任务——猜出了我在想的生物或是元素是什么，并且通过问只能回答是或否的问题完成了任务。计算思维是从分解开始的，在这一步中，你把大问题分解为一个个的小问题。在第一项活动中，你仅问了 3 个问题就指出了我想的是 8 种中的哪一种。

如果让你猜出我在 16 种生物中想的是哪一种，每问一个问题排除一般选项，你需要问几个问题呢？不妨来画张表看一下。

如果每个问题能排除一半选项：

- 第一个问题排除了 8 个后还剩 8 个
- 第二个问题排除了 4 个还剩 4 个
- 第三个问题排除了 2 个还剩 2 个
- 第四个问题再排除 1 个就只剩下了最后 1 个

所以对于 16 个选项来说，每个问题能排除一半选项的话，我们只需要 4 个问题，就能得出答案。所以对于 32 中生物我们需要问多少问题呢？

5 个问题

是的，每个问题有两种可能性。“是”和“否”。对于：

- 8 种可能性 $2*2*2=8$
- 16 种可能性 $2*2*2*2=16$
- 32 种可能性 $2*2*2*2*2=32$

在多项实验中查看数据时，识别是否存在模式、存在什么样的模式对找出潜在的原因或原理是非常有帮助的。

假设地球上有 8 亿种生物。每问一个问题排除一半选项，你需要多少个问题能猜出我想的是哪种生物呢？

这将需要 33 个问题，想要从上亿种可能性准确的找出答案，这个数字看起来也许有点小。20 个问题的游戏可以从超过 100 万种可能性中猜出答案，这对于大多数的物体来说都应该是足够了。

现在，你可能搞清楚了，你需要提问的次数可以通过计算所有可能性的 \log_2 对数得到，或者通过计算 2 的几次方接近或等于可能性总个数： $2^4=16$ ， $2^5=32$... $2^{33} \approx 8.5$ 亿。这种从实验中总结出规律，并上升为一种普遍规律、公式或者定律的方法，就是通常所说的**抽象法**。

在此次学习中，通过使用计算思维的步骤，你学到了理论上你需要多少问题来对地球上的生物加以分类。计算思维不仅让我们得到了答案，而且明白了内在的，可以迁移到其他新问题中使用的基本规律及计算过程。

如果你对此想有更多了解，可以搜索以下信息：信息理论及决策树。[可汗学院](#)有关于信息理论的视频，[不插电的计算机（CS Unplugged）](#)有关于检索和分类算法的活动，也有这次活动的数

库和信息理论。

3-2 课程反思

下面是本章所讨论的、可以从中找到计算思维过程的一些活动总结。计算思维是一个高度跨学科的内容，你可以在任何其它学科中找到相关的活动。

数据压缩

- **分解** - 将一整张照片视为许多像素点来处理。
- **模式识别** - 应用不同的位掩码，以了解不同的位掩码对图片所需存储空间和图片质量的关系。
- **抽象化** - 使用发现的模式来决定使用合适的位掩码，既能够压缩图片体积也能够保持合适的图片质量。
- **算法开发** - 尽管在本章节中你并没有开发任何算法，你也许已经认识到位掩码的不同属性可以帮助你改善算法的使用。

音乐

- **分解** - 观察歌曲的不同部分并将其划分为不同的组成，例如：节拍，音符，以及和声，然后分别检查这些组成成分。
- **模式识别** - 调整每个变量，诸如音符的长度以及音阶中音符的频率，来找到一个你最喜欢的设置。
- **抽象化** - 添加第三个钢琴来营造和声。
- **算法开发** - 尽管在本章节中你并没有设计任何算法，使用已经完成的代码来调整设置，使用另一种音阶，以及添加额外的钢琴来得更加轻松使你能够进行探索。你也许有一些关于音乐其它属性的不同想法，并尝试将它们应用到以后对音乐的改进之中。

小海龟几何

- **分解** - 将一个任务转换为一系列步伐和转向，例如画一个正方形。
- **模式识别** - 观察重复步伐和所画图形的边之间的相互关系。
- **抽象化** - 使用转向角度和步伐的数目中的模式来深入了解步伐数目，角度和多边形边的数

目之间的关系。

- **算法开发** - 设计一个算法用来绘制任意形状的多边形。我们有很多方法来拓展该算法。例如，当计算小海龟的每一步应该多长的时候将边数考虑到。

分类

- **分解** - 观察生物体的不同的分类方式。例如：生物体是水生生物或者是有翅膀能在天上飞。
- **模式识别** - 研究一种方法用以有效地根据生物体的共同特征来进行分类。
- **抽象化** - 测定使用该方法的情况下需要多长时间来进行分类，有助于你了解对所有已知生物体进行分类需要多少问题或分级。
- **算法开发** - 在本章节中你并没有开发出一个算法，但是你可能已经想到了如何更有效的对生物进行分类。

现在你已经大致体验过计算思维的过程了。在你的课程表中，是否还有其它课题能够从学生应用计算思维中获得帮助呢？你可以在课程社区中分享你的观点。

四、算法开发

4-1 概述

在本节中，你可以尝试体验一些稍作修改就能应用于课堂教学上的活动。这些活动会涉及到算法开发，以及编辑一段已有的代码或增添新的代码，但是并不要求提前对编程有所了解。这些活动中都涉及到计算思维过程，其重点在于算法开发。请点击下面的活动的链接来开始体验吧。你也可以随意尝试其他领域的活动。像你对学生所期待的那样，探索、试验、游戏和享受活动吧！

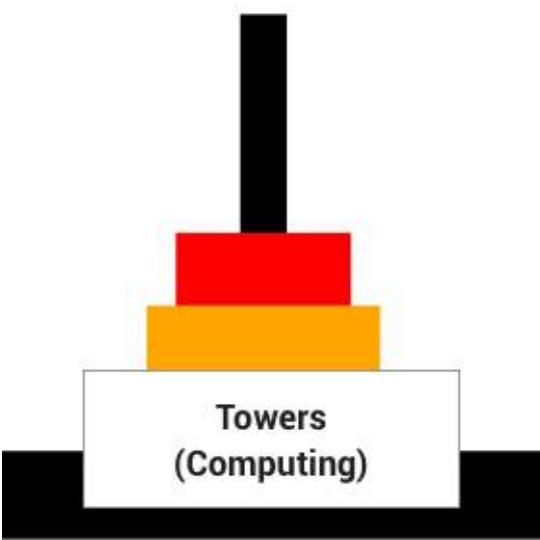


图 4-1 汉诺塔案例示意图

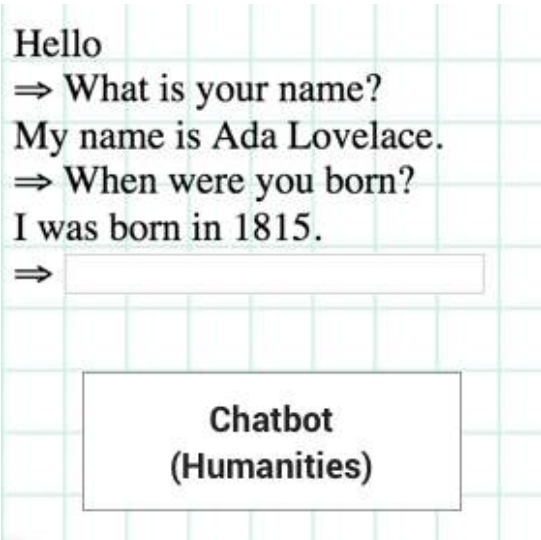


图 4-2 聊天机器人案例示意图

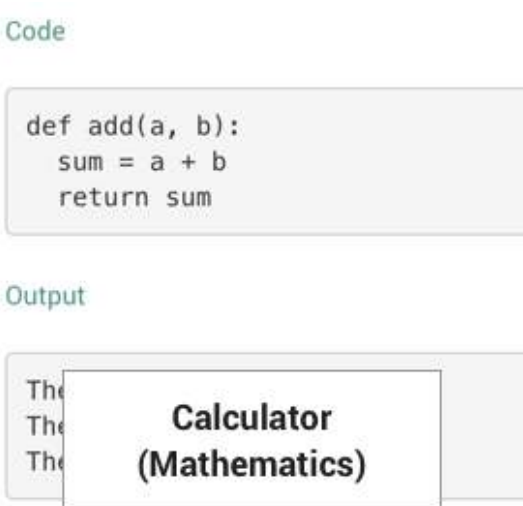


图 4-3 计算器案例示意图

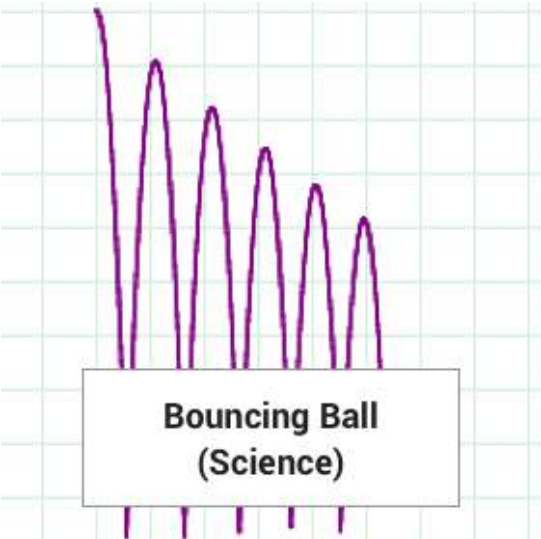


图 4-4 弹球案例示意图

4-1-1 计算机科学：汉诺塔

传说在一个房间里有三根柱子（塔）。在最左边的塔上有 64 个金制的圆盘摆成一垛，从上到下按照从小到大进行排列（图 4-5 中是一个只有 3 个圆盘的简化版本）。负责看管并移动圆盘的人被告知，当所有的圆盘从一个柱子移到另一个柱子上时，世界末日就会降临。

移动圆盘时要遵守的规则：

1. 一次只能移动一个圆盘，并只能从一根柱子的顶端移到另一根柱子的顶端。
2. 较大的圆盘必须位于较小的圆盘下方。（换句话说，塔上的圆盘永远保持金字塔型）

如果一个人每次移动圆盘都要花 1 秒的时间，你认为他要花多久才能解决这个问题？

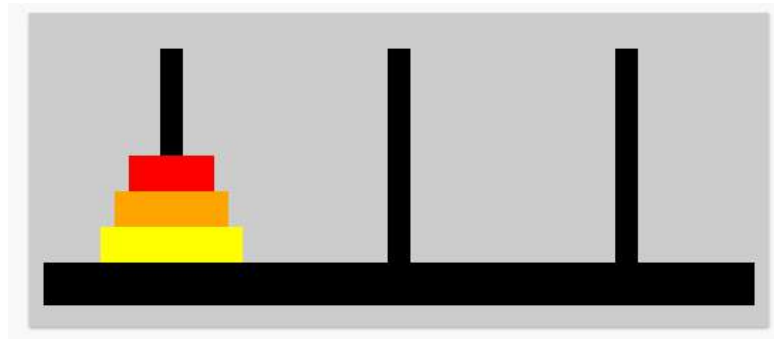


图 4-5 3 个圆盘的汉诺塔

你可以直接尝试使用 64 个圆盘来解决这个问题，你也可以尝试将这个复杂问题简化成一个比较简单的任务，看看是否可以在指定圆盘的数量下，使用最少的移动步骤来解决这个问题，或者，你也可以试着猜猜看。在计算思维里面，这个过程叫做“任务分解”。

在上面的汉诺塔的例子中，在左边的塔上有 3 个圆盘，试试看，如何将这 3 个圆盘移到中间或者右边的塔上。

同样，在尝试解决这个问题的时候，必须遵守下面的规则：

1. 每次只能移动一个圆盘，而且只能从最上面的圆盘拿起并移动到另外一个塔的上面
2. 任何时候，任何一个塔上面的圆盘不能比它下面的圆盘大。

在你解决了或者尝试解决了下面的问题以后，试试看如果有 4 个圆盘，你该怎么办？

根据你前面的尝试，对于 4 个圆盘的问题，估计你最少需要移动多少步才能解决这个问题？

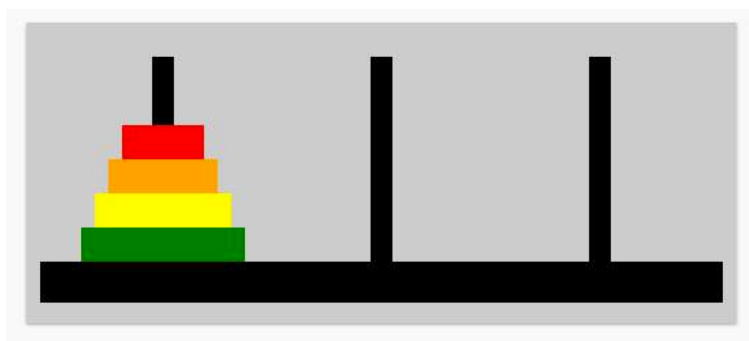


图 4-6 4 个圆盘汉诺塔的操作界面

当你将圆盘在不同塔之间移动的时候，你有没有发现一些规律，即便你可能还说不清楚具体规律是什么（有或没有）。

最少的移动步骤是：

- 3 个圆盘的最少移动是 7 次
- 4 个圆盘的最少移动是 15 次。

基于这些实例的解决方案，你认为它能解决汉诺塔问题中的 5 个圆盘的移动吗？

汉诺塔这个问题的最少移动次数遵守这个规律： $2^n - 1$ 。这里 n 是圆盘的数目。比方说，如果有 5 个圆盘，那么最少需要的移动次数就是 $2^5 - 1 = 31$ 。

下面是一个有 3 个圆盘的汉诺塔难题的移动顺序：

1. 塔 A，塔 C
2. 塔 A，塔 B
3. 塔 C，塔 B
4. 塔 A，塔 C
5. 塔 B，塔 A
6. 塔 B，塔 C
7. 塔 A，塔 C

根据上面移动的列表，在第几次移动的时候开始重复？答案是在第四行移动的时候开始重复。

移动三个圆盘的例子可以看出：一个圆盘在第四步在 A 和 C 塔间移动，并在第七步开始重复。从这里你可以发现三个对于你的**算法**有用的信息。

通过观察移动的规律，我们发现：

1. 在 A 和 C 塔间的移动是第一步，因此应该被放在移动的第一步。
2. 由于 A 和 C 塔间的移动在第四步和第七步重复，你应该需要重复这些步骤直到问题解决。

3. A 塔和 C 塔的移动是第四步，因此，必定在第四步到第七步之间还得有 2 个移动。

根据上面的这些规律，你可以建立你的算法了。

解决一个有 3 个圆盘的算法是：

1. 在 A 和 C 塔间移动圆盘
- 2.
- 3.
4. 重复这个过程直到问题解决。

根据上面 3 个圆盘的汉诺塔难题的移动顺序，塔 A 和塔 C 的步骤重复之后，下一步是哪两个塔？（是塔 A 塔 B，塔 A 塔 C 还是塔 B 塔 C？）

是的，这就是模式中需要重复的步骤，你可以把它添加为算法的第二步。

解决 3 个圆盘的汉诺塔的算法如下：

1. 在塔 A 和塔 C 之间移动一个圆盘
2. 在塔 A 和塔 B 之间移动一个圆盘
- 3.
4. 重复此过程直到问题解决。

太好了，我们的算法马上就要做好了。

解决一个有 3 个圆盘的例子是：

1. 将圆盘在 A 和 C 塔移动
2. 将圆盘在 A 和 B 塔移动
3. 将圆盘在 C 和 B 塔移动
4. 将圆盘在 A 和 C 塔移动
5. 将圆盘在 B 和 A 塔移动
6. 将圆盘在 B 和 C 塔移动
7. 将圆盘在 A 和 C 塔移动

你能找出最终的模式并写成算法吗？在上面的例子当中，最后重复的塔是什么？（是塔 A 到塔 B，塔 A 到塔 C 还是塔 B 到塔 C？）

恭喜你，你已经发现了这个问题的规律并将它抽象成为一个算法。

解决一个有 3 个圆盘的算法是：

1. 在塔 A 和塔 C 之间移动圆盘

2. 在塔 A 和塔 B 之间移动圆盘

3. 在塔 C 和塔 B 之间移动圆盘，重复这些过程直到问题解决

事实证明，只要圆盘的数目是奇数（3，7，13），就可以用上面的办法来解决这个问题。将这两个算法结合，只要有足够的时间就可以解决任何版本的汉诺塔问题。

现在我们再重新看一下原始的问题：如果某人解决一个有 64 个圆盘的汉诺塔的问题，并且他每秒钟移动一个圆盘，当他解决这个问题的时候，世界将会毁灭。你觉得这个事情会多久发生？

根据你前面发现的规律，解决这个问题需要多长时间？（1 天？100 年？或者几百万年？）

你会发现，解决这个问题的时间随着圆盘的数量增长而急剧的增加。可以用 $2^n - 1$ 这个公式来推算解决有 n 个圆盘的汉诺塔需要的时间，如果有 64 个圆盘，需要的时间是 $2^{64} - 1$ ，这个数字是一个超级大的数字，如果每一秒移动一个圆盘，解决这个问题需要的时间甚至超过了宇宙的年龄，现在，你大可放心了！

这个例子很有名不仅仅是因为它的算法和趣味，其重要性在于它是一个递归的绝佳例子。递归算法是计算机科学专门用来描述一个算法本身的问题，所以接近作为一系列较小的问题来解决。

你不愿意看看你是如何通过递归运算来解决汉诺塔这个问题的？如果你愿意在这个方面深入钻研的话，后面将会有更加让人兴奋的发现。

你已经有了一个解决汉诺塔的算法。现在你可以将计算思维再次使用一次以开发出一个递归算法。

在进入这个汉诺塔问题之前，再看一下使用乘法运算进行递归运算的一个例子。当你把乘法运算当作一个重复的加法运算的时候，你就可以构造出一个递归的算法。

任务：将两个数 a 和 b 相乘

1. 规则 1：如果 b 等于 1，那么结果就是 a

2. 规则 2：将 a 和 $b-1$ 两个数相乘，然后将乘积加到 a

例子： $5 * 4$

1. $(5 * 3) + 5$

2. $((5 * 2) + 5) + 5$

3. $((((5 * 1) + 5) + 5) + 5) + 5$

4. $5 + 5 + 5 + 5$

5. 答案是 20

这看上去挺复杂，一个递归运算有的时候看上去更简洁。例如当你使用某些算法，如排序数字

或者产生一个序列数字的时候。

在前面我们重复提到的汉诺塔算法步骤中，有些步骤被不停的重复直到问题得到解答。然而，除了这个办法，还有另外一个更通用的办法，那就是不断将比较小的圆盘移动到其它塔上直到你可以移动最大的那个圆盘。

因此，使用递归运算解决这个汉诺塔的算法可以被描述为：

任务：将所有的圆盘从 **A** 塔移动到 **C** 塔

1. 将除了最后一个圆盘的所有的其它圆盘从 **A** 塔移动到 **B** 塔
2. 将最后一个圆盘从 **A** 塔移动到 **C** 塔。
3. 将所有的圆盘从 **B** 塔移动到 **C** 塔。

如果你仔细观察的话，你就会发现，你一个移动步骤其实都需要一组更小的算法步骤来实现。维基百科中关于“汉诺塔的递归算法”可以帮助你理解这个原理。

如果你对于这个话题还有进一步的兴趣，你可以搜索排序算法，递归，汉诺塔游戏来进一步了解。这项活动的潜在标准可以与学生一起使用。

4-1-2 人文学科：聊天机器人

在本活动中，你会通过应用计算思维过程来创建一个聊天机器人程序。

你也许想知道什么是聊天机器人？一个名叫阿兰图灵的数学家曾经预测过，未来某一天计算机将能够完成高度复杂的任务。他还提出了智能阈值的概念。

这个测试被称为图灵测试。这是一个十分直白的测试：如果一台计算机能够和人类进行交流，而人类无法分辨出跟他进行交流的是一个人还是计算机的话，该电脑就被认为通过了图灵测试而拥有了智能。



图 4-6-a 世界上第一个程序员阿达·洛夫莱斯

该聊天机器人是基于世界上第一个计算机程序员阿达·洛夫莱斯（她出生于 1815 年，她的父亲是一个诗人，她的母亲单独抚养阿达长大，并把她培养成了一名数学家和科学家）。阿达编写了第一个计算机程序。该程序运行于一台早期的机械计算机分析机。

在 1843 年的时候，27 岁的阿达在分析机上翻译了一篇文章，并在其中添加了她自己的笔记。在这其中包含了一段被认为是世界上第一个计算机程序的内容。程序是一连串指令的集合，这些指令会被机器所执行。阿达曾经预测过像这样的机器可以被用来创作音乐，画图，以及其它不仅仅是用来做计算的功能。她在计算机真正实现这些功能的一个世纪前就有了这样的想法。她经常将诗歌和科学相结合，认为想象力和直觉对于数学和科学有着非常重大的作用。

当阿达写下这篇著名的关于分析机时，她已经和洛夫莱斯伯爵结婚并有了三个年幼的孩子。令人悲伤的是，阿达的一生都被疾病所困扰，最终年仅三十六岁的她在 1852 年死于子宫癌。

在被阿达了不起的事迹所启迪之后，你可以开始创造一个会像阿达本人一样回答问题的聊天机器人。

如图 4-6-b 是阿达的初始聊天机器人。当你运行这个代码的时候，一个用来输入你的问题的文本框会出现。目前为止，“她”可以回答的问题只有她的名字和她的出生日期。如果你输入问题“你是谁？（Who are you?” 或者“你生日是什么时候？（When were you born?”）你就可以看得到答案。

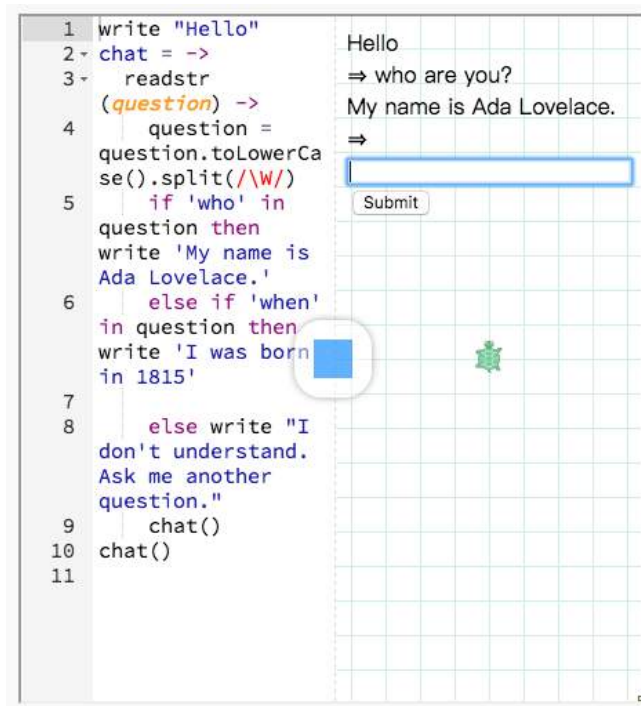


图 4-6-b 阿达的初始聊天机器人

互动链接: <https://jsuansiwei.pencilcode.net/edit/HumanitiesChatbot>

假设你是一个对阿达（Ada）一无所知的人，你会问她什么问题？如果你必须以阿达的身份回答这些问题，她会知道哪些事情以及她会如何回答这些问题？你可以根据前面提供的信息来设计答案，或者自己做一些相关研究。

模仿一个人类是一项十分困难的任务。首先，你可以将其分解成若干个容易处理的任务。你认为人们会问阿达哪些问题？

这个聊天机器人使用了逻辑语句 **If** 和 **Else**。下面是算法的具体步骤：

1. 用户键入一个问题。
2. 算法检查用户名是否匹配，如果（**If**）用户的回答和数据库匹配，就（**Then**）给出一个答复
3. 否则（**Else**），计算机就给出“我不理解你说的是什么”的答复。

你可以随意添加或者更改现有的答复的选项。

条件语句 **If** 和 **Else** 非常有用，基本上每一个软件的重要功能都会使用到这个语句。

利用现有的代码作为一个指导，在第 7 行添加新的 **If/Else** 语句，当你问“你是哪里出生的时候”，阿达聊天机器人可以回答“我是在英国伦敦出生的”。当你根据已经有的例子添加了一个新的问题以后，点击播放按钮，重新运行代码。

除了逻辑语句 **If** 以外，你还可以添加布尔值逻辑量让聊天机器人可以测试在某个单一条件下，某两件事是否真实的。聊天机器人可以回答诸如“你是谁”这样的问题，但是如果你想加入“你叫什么名字”这样的问题的时候，你应当可以针对“什么”加入一个 **If** 语句，但是最好算法可以自动搜寻“名字”这个关键词。

不同于通过在算法中添加一个同样答复的算法，代码应该可以自动检查问题是否包含诸如“谁”或者“名字”这样的关键词。现在你可以通过不同的方式来问阿达的名字了。

编程语言是一门像其它语言一样，有一定语法规则。事实上由于计算机不能接受自然语言测试，因此你在使用编程语言的时候必须很小心你的输入代码必须满足那些规则。尽管如果一封邮件有一些小的语法或者拼写错误，你还可以理解这封邮件。但是计算机就不是这样了，它仅仅会执行你让它做的，哪怕它是错误的。使用不正确的语法和拼写错误会造成计算机错误。或许未来的某一天计算机可以能够觉察到你的一些细小的失误并认识到你真实的意图，但是在那天来临以前，我们必须按照规则和算法编写我们的程序，并检查程序中可能的错误，以避免失误。

你可能会继续让你的聊天机器人在许多方面继续重复回答问题。但是如果你这样做并继续添加问题，你的程序也会变得越来越长。

你可以上网查看其它聊天机器人的例子，你会发现其中绝大多数都和我们的阿达机器人有着类似的逻辑基础。他们都会检查特定的内容或者模式，然后应用对应的算法。如图 4-6-c 是一个聊天机器人，它能在遇到它无法回答的问题时进行学习，并在内存中记住问题和对应的答案。

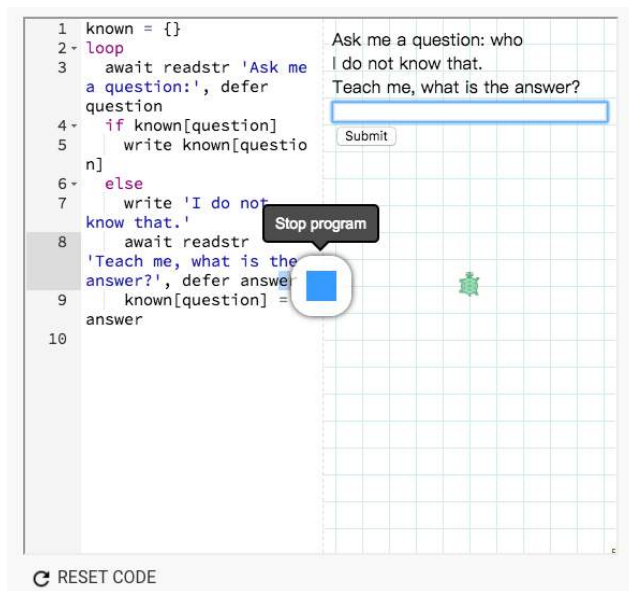


图 4-6-c 一个聊天机器人的例子

试着运行这段代码，然后回答一些问题。在你教会机器人一个新问题之后，它应该就能回答这个问题了（前提是你得用同样的方式问这个问题）。

这种通过训练编写算法而非提前列出每种可能性的过程，便是机器学习的一个范例。谷歌正利用机器学习来更好地理解模糊检索和网络上的很多其它实例中。然而，算法学习和有机体（比如人类）学习之间的差距仍然存在。为何不反思一下这些差距，同时想想看作为一个有生命和智慧的存在意味着什么。如果有学生参与其中，还有很多标准可供设置。

4-1-3 数学：计算器

科技时常可以迅速而高效的为你提供许多问题的答案，诸如：为什么天空是蓝色的？我朋友的生日是什么时候？ π 的值是多少？仅仅可以用来回答诸如天为什么是蓝色的或者画出 x^2-1 的图像是不够的。科技真正的能力在于你把它用作创造的平台。

使用计算器是一回事，但是想象一下你也许可以制作一个你自己的计算器！你认为你的基础计算器应该拥有哪些功能？

好了，我们先从加法开始。你准备如何向一个儿童解释加法的流程？

在数学中，和计算机科学一样，一个函数表示的是当将一组变量输入到一组公式中的时候，可

以对应的一组输出。如 $F(x)=x^2$ ， $F(3)=9$ ， $F(10)=100$ 。

对于计算器来说，你可以创建一个叫 **add** 的函数，这个函数可以输入两个输入值，并且一个输出值用来得到这两个输入值的和。如：**add(3, 5)→8**

下面是一个加法算法的例子

1. 使用 2 个数字作为输入
2. 将这两个数加在一起
3. 输出结果。

在 **Python** 编程代码中加法算法的例子如下，尝试使用不同的输入数字输入到函数中，看看代码运行以后，结果是什么。

Language: python

```
1 def add(a, b): # Two inputs
2     sum = a + b # Add numbers together
3     return sum  # Output the result
```

RUN

```
def add(a, b): # Two inputs
    sum = a + b # Add numbers together
    return sum # Output the result
```

Output

```
The sum of 1 + 1 = 2
The sum of 2 + 3 = 5
The sum of 10 + 10 = 20
```

图 4-7-a 加法函数的例子

如图 4-7-a 是一个加法函数的例子。

现在，每当语句 `add(a,b)` 被使用的时候，你应当可以看到结果是 `a+b`。尽管在计算的时候你使用的是 `+` 这个运算符号，但是在函数中这样表示 `(a,b)` 是可以接受的。

那么乘法是怎么通过函数实现的呢？你可以使用一个叫 `multiply(a, b)` 的函数，这个函数将两个输入值相乘并将结果返回出来。想想看，你是否可以使用已知的加法函数来实现一个乘法的功能？

想想看，你准备如何解决上述问题，请继续往下看。

对于计算器来说，创建一个乘法函数的方法是使用递归。这个概念在数学和计算机科学中很受关注。递归是一个和它自己相关的算法，递归可以让一个复杂的问题分解成一个个小的容易问题并加以解决。递归是计算机科学里面的一个算法。

斐波那契数列是数学中递归的著名例子。1, 1, 2, 3, 5, 8, 13.....每一个数都是前面两个数相加的结果，公式如下： $F(n)=F_{n-2}+F_{n-1}$ 。举例来说，最开始的两个数 1，1 的和就是 2（第三个数），而第二个数 1 和第三个数 2 相加就是第三个数 3。以此类推。这个规律当你需要开发一个和乘法相关的算法时就非常有用。

下面是一组使用 `add`（加）的功能来表现乘法规律的例子。

任务：将两个数 `a` 和 `b` 相乘。

规则 1：如果 `b` 等于 1，那么答案是 `a`。

规则 2：将 `a` 和 `(b-1)` 相乘，并将结果输出给 `a`。

例子：5 * 4 的乘法例子。

1. $(5 * 3) + 5$
2. $((5 * 2) + 5) + 5$

3. $((5 * 1) + 5) + 5 + 5$

4. $5 + 5 + 5 + 5$

5. 结果是 20

在递归函数中，需要一个“初始状态”，这个初始状态是用来让函数停止运行的。规则 1 就是一个初始状态。因为乘法单位是 1，当 $b = 1$ 的时候，这个函数就应该停下来。在前面的例子中，你重写了指令，并将这些指令放入函数中。为什么不基于你的算法而定制一个你自己的函数？

任务：将两个数 a 和 b 相乘。

规则 1：如果 b 等于 1，那么答案是 a 。

规则 2：将 a 和 $(b-1)$ 相乘，并将结果输出给 a 。

在下面 Python 编程代码中我们可以看到一个乘法算法的例子。改变输入的数值，尝试运行一下程序，看看结果有什么不同。

Language: python

```
1 def add(a, b):  
2     sum = a + b  
3     return sum  
4  
5 def multiply(a, b):  
6     if b == 1:  
7         return a  
8  
9     if b > 1:  
10        return a + multiply(a, b-1)  
11
```

RUN

```
def add(a, b):  
    sum = a + b  
    return sum  
  
def multiply(a, b):  
    if b == 1:  
        return a  
  
    if b > 1:  
        return a + multiply(a, b-1)
```

Output

```
The result of 1 * 1 = 1  
The result of 3 * 5 = 15  
The result of 10 * 10 = 100
```

图 4-7-b 乘法函数的例子

这个一个使用递归运算来构造一个乘法的例子。

如果“初始状态”被去除，递归的结果会发生什么？结果是这个指令将会一直运行下去。那么为什么计算机自己不去发现这个错误呢？

这是一个**停止问题**。你可以使用一个函数，看看需要多长时间才能够结束。但是实际上计算机不是总是知道它需要在哪里停止。在解决这个问题之前，在编写算法时你必须格外小心。

根据你在前面例子里的结果，尝试用递归来完善除法函数的缺失部分。

```
1 def subtract(a, b):
2     difference = a - b
3     return difference
4
5 def divide(a, b):
6     # Fill in the base case
7     ''' Replace 1 below
8         with your code'''
9     quotient = 1
10    return quotient
11
```

RUN

以下是补充完整的除法算法及输出结果：

```
def subtract(a, b):
    difference = a - b
    return difference

def divide(a, b):
    quotient = queo(a,b,0)
    return quotient

def queo(a, b, c):
    v=subtract(a,b)
    if v< 0:
        return c

    if v >= 0:
        return queo(v,b,c+1)
```

Output

```
The quotient of 4 / 1 = 4
The quotient of 6 / 2 = 3
The quotient of 100 / 10 = 10
```

You did it!

图 4-7-c 除法函数的例子

递归不是一个容易理解的概念。如果碰到了问题，不要灰心丧气，可以尝试将它写出来，这样可以帮助你问题视觉化并实验结果。

现在你可以利用你前面的所学准备创建一个计算器了。可以试一下这个函数 $\text{square}(a) \rightarrow a^2$ or

$\log(a) \rightarrow \log_{10} a$ 。

建立自己的计算器是计算思维的一个很好的例子，在这过程中你会遇到一个较大的问题，并将之分解成较小的部分，如函数。当你开发了乘法算法，并能将算法推广到新算法例如除法中，你就可以做到这一点。

如果你想更深入地探索这个主题，你可以在互联网上搜索递归和函数编程。**Bootstrap** 课程通过编程教授学生代数和几何概念。如果让学生使用此课程，可与这项活动的潜在标准配合使用。

在技术不可用的情况下，学生可以写出算法，告诉某个人输入什么数，轮流尝试，测试算法设计。

4-1-4 科学：弹力球

当如今的科学家、工程师、动画师以及其他需要建设模型和进行仿真来预测和模拟真实世界的时候，他们常常会使用到物理学。

在本活动中，我们将会使用小海龟来模拟一个弹力球。我们会通过提供正确的一系列指示（算法）来训练小海龟进行移动和反弹。

如图 4-8-a 的代码会每秒检查 100 次鼠标的位置，并将小海龟的方向指向该位置。

移动你的鼠标指针在小海龟附近环绕。你将会看到小海龟转向你的指针的方向。

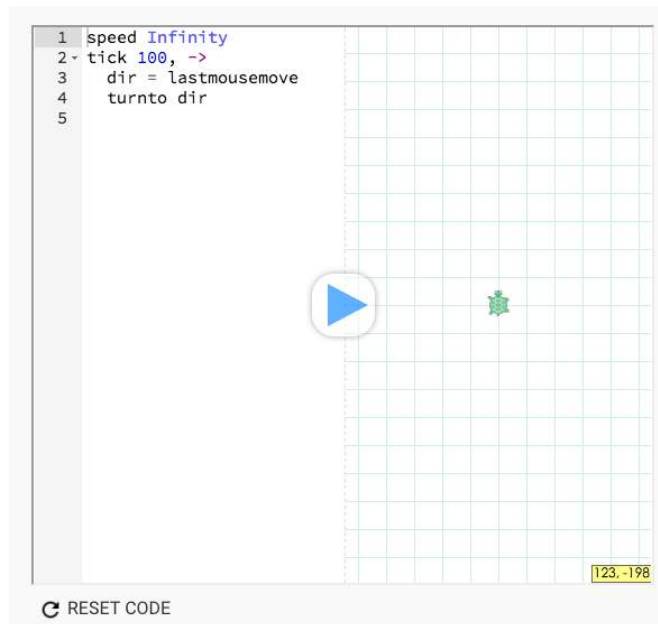


图 4-8-a

互动链接：<https://jisuansiwei.pencilcode.net/edit/ScienceBouncingBall>

你准备好让小海龟移动了吗？现在，如果我们想要小海龟向我们指针的方向移动的话，应该在

我们的仿真中增加什么信息？

但是我们已经知道了小海龟的方向，因为这是基于鼠标的最后位置决定的（`last mousemove`）。为了让小海龟真正朝这个方向前进，我们需要给出哪些其他信息呢？

对于我们的小海龟来说，它要在特定的方向上随着时间移动一定距离，这意味着需要为模拟添加**速度**。

如图 4-8-b 修改后的代码中，我们设定小海龟每次移动的距离为 1（第三行），并且增加小海龟向前移动的指令（第六行）并重复 100 遍。

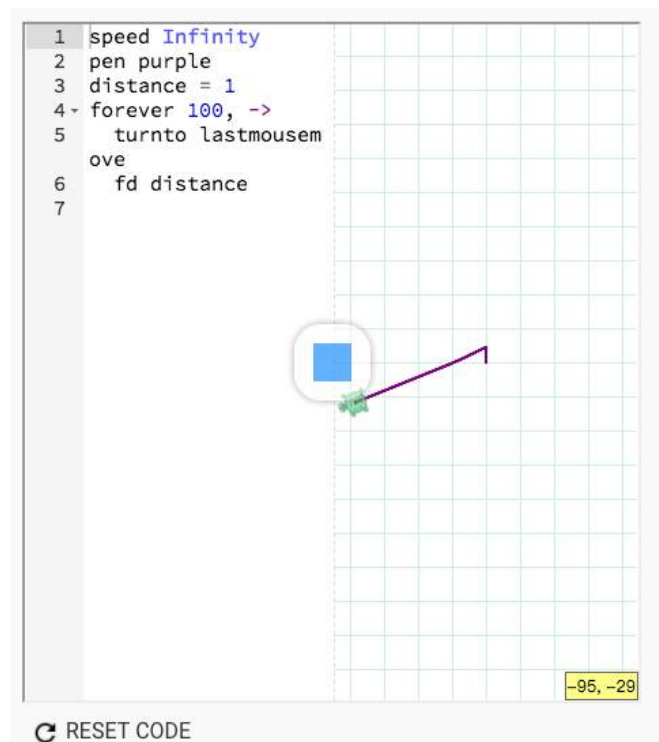


图 4-8-b

因此，随着时间推移，小海龟在特定方向上随时间移动一定距离（即速度）。在物理学中，可写为：速度=距离/时间。

点击播放按钮，像以前一样移动鼠标，但当小海龟跟着鼠标移动时，仔细观察。我们在小海龟的肚子上贴了一支笔，因此你可以看到它走过的路径。

让我们教小海龟跳跃。别担心，不会受伤！

当一个小球掉落时，每秒会发生什么，你会如何向一个小孩子解释呢？花一点时间把你解释的指令清单写下来。没必要包含方程式。如果你愿意，画一幅图，说明当小球下落时每秒会发生什么。随意使用自己的话描述，但尽量具体描述，并将其分解为步骤。

当你想到最佳的指令清单时，继续往下看。

你的指令也许看起来不一样，但是我打赌我们可以让这只小海龟在任何时间弹跳。

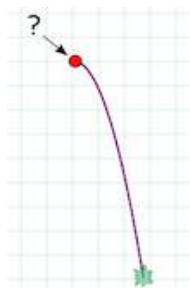


图 4-8-c

如图 4-8-c 小海龟在开始下降之前的瞬间向下移动的速度（初始垂直速度）有多快？

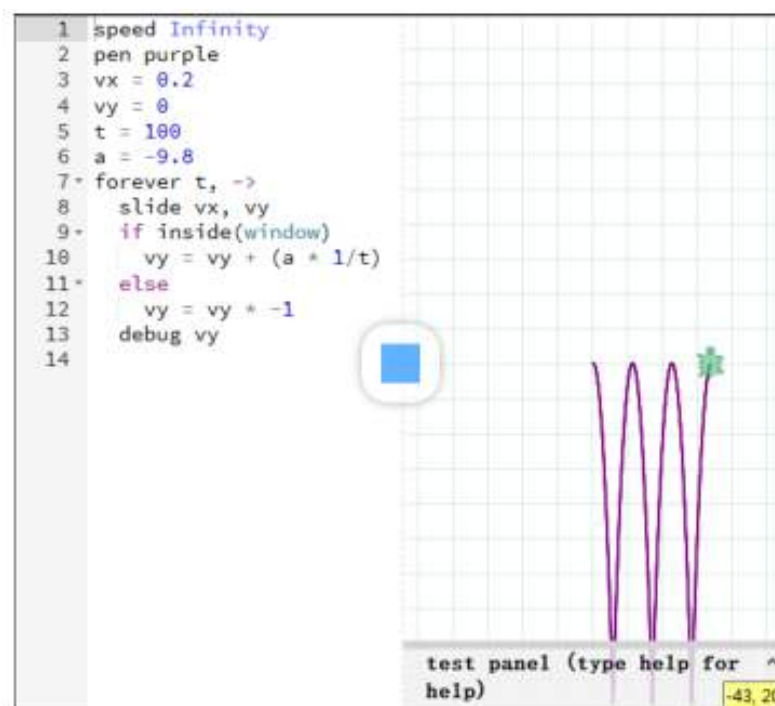


图 4-8-d 弹力球的代码与运行界面

如图 4-8-d 弹力球的代码与运行界面，在它开始下降之前，它根本不应该移动。因此，模拟器将初始向下的速度设置为 0。

我们来看一下这段代码：

第 3 行：小海龟 x 方向的速度

第 4 行：将小海龟 y 方向的速度设为零

第 6 行：重力加速度。

第 7 行：循环每秒重复 t 次。

一旦模拟器开始运行，小海龟将向下移动，直到到达窗口底部（地板），然后它会安全地弹回到顶部的起始位置。

问题：当到达窗口底部时，你觉得哪行代码改变了小海龟的方向？

在这个例子中，因速度为负，小海龟向下移动。哪一行代码会使速度变为正数？

第 12 行代码，当小海龟到达窗口的底部时，小海龟的速度乘以 -1 ，逆转了方向。

如果你熟悉描述物体降落的物理方程式，你可能会惊讶地看到第 10 和第 12 行代码。这些模拟旨在模拟现实，我们可以通过修改方程式（如果你曾试图在现实中改变重力，你会意识到模拟是多么有用）进行实验。

我们的小海龟正享受着美好时光，感受跳跃的喜悦。耶！你可以随意摆弄各种数字和代码本身。如果你想挑战自己，可以让小海龟在每次跳跃中失去一点点体力，这样也更加现实。

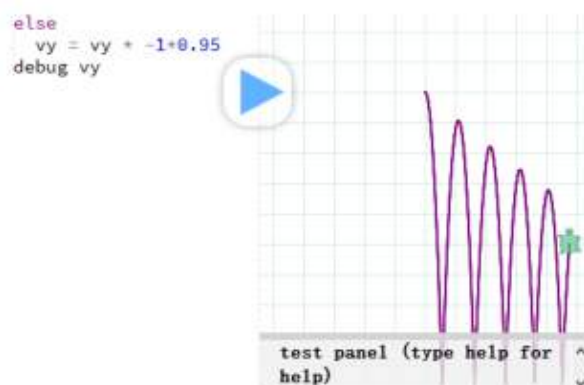


图 4-8-e

有很多方法可以实现这一点，你可以尝试搜索【恢复系数】一词，作为一个开始。

有关这些类型模拟的更多示例，你可以查看 [PhET](#)。这项活动的潜在标准可以与学生一起使用。

4-2 课程反思

这些活动都基于你在模式发掘章节中关于计算思维的经验，并且都强调了算法开发。假如你不能完成这些算法，这没关系，因为更重要的是你能够观察了解到计算思维的过程和可能性。你会发

现经过不断的练习之后，这会变得更加容易和自然。而且在本课程中有一个教育者网络社区，他们在不断尝试将计算思维应用到他们的课程当中。你可以随时在该社区中发表问题并且继续你的学习。

汉诺塔

汉诺塔问题是一个在计算机出现之前就很流行的算法案例。尽管这是一个很有趣的谜题，它也是一个非常有名的递归和算法效率重要性的例子。随着碟子的数目以线性增加，问题解答步骤的数目以指数增加。在如今，我们在网络和商业交易中使用的算法必须能够处理数十亿用户在一秒内产生的数据。因此，计算机科学家们一直在寻找能够降低算法复杂度而提高算法质量的方法。

聊天机器人

人工智能有可能诞生吗？计算机是否能够和你进行一段对话并不让你意识到它的身份？当计算机在扮演一个人的时候，他需要多少信息呢？我们之所以在活动中选择该算法，是因为他有着相对直观的逻辑。计算机科学家们在训练他们的算法拓展连接的层级和网络的时候，很像我们自己的大脑在学习新事物的过程。这些算法通常训练用来完成特殊的工作，例如识别一个目标或者解析口语命令。图灵为人工智能设计的测试已经被一个算法通过了，但是这是不可避免的？或是不可能的？

计算器

科学技术和互联网使获得海量的信息成为可能。以往要数个小时才能找到答案的问题，现在只需要几秒钟就能解决，而很复杂的数学方程只要输进计算器就能很快被解答出来。尽管有些人禁止在他们的课堂中使用这些技术，但是他们其实有着另一个选择。在本活动中，学生可以自己为课堂创作使用的工具，你有机会为一个计算器来开发算法。使用计算器是一回事，但是为计算器开发一个算法则可以让你更好的理解数学中潜藏的模式。当学生可以清晰的表达和彻底理解数学过程的时候，他们就获得了非常深刻的对数学本质的了解。这样，即使在很久以后学生已经遗忘了具体的数学公式，他们也可以将这些理解应用到新的问题之中。

弹力球

实验室是科学家进行试验来验证他们的假设的地方。对某些研究来说，在传统实验室中进行实验是非常困难的，因为它们的研究对象太过微小（例如亚原子微粒），或者需要一个跟地球完全不同的实验环境（例如宇宙中的黑洞），又或者需要极端的环境（例如飞机的测试）。数十年来，科

学家和工程师们通力合作，创造了许多模型和仿真系统来测试他们的理论和原型机。在本活动中，你模拟了一个小球的弹动，并且不断改进这个模型使得它越来越真实。没有任何一个模拟是完美无缺的，但是只要算法不断改善，仿真就可以帮助我们为一些复杂的现象提供预测，例如全球变暖。科技的成本和仿真软件以及编程语言的可获得性，使得在全球各地进行科学研究称为可能。

在第一节中，你应该已经仔细考虑了那些你的课程中可以应用算法的领域。在你进入课程项目之前，你认为是否有一些新的课题，在其中学生们可以学习到如何应用计算思维和算法开发？你可以在课程社区中分享你的观点和课题。

五、应用计算思维

5-1 项目概要

目的：使用从本课程中学习到的技能，来更有效的将计算思维和你的课堂教学相结合。

第一部分：在项目的第一部分，你将会书写一份陈述来描述如何将计算思维应用到你的领域中。

第二部分：在项目的第二部分，你将会记录下你是如何计划将至少一个计算思维的概念整合进入你的课堂、活动、单元、项目、模型或者课程中。

样例项目：

- [整合计算思维样例课程计划：探索计算思维课程计划](#)（或者参考附件一）
- [交流计划案例](#)（或者参考附件四）
- [策略计划案例](#)（或者参考附件五）

5-2 反馈、评估、评分

该项目能使你自己受益，同时我们认为巩固知识最好的办法就是不断练习。

如果你想要了解其他学习者的项目，欢迎你在 [class Google+ Community](#) 上查看。

5-2-1 项目，第一部分

第一部分：书写一份陈述来描述你的领域中应用计算思维。

提示：

- 在该项目的第一部分中，书写一份陈述来描述如何将计算思维应用到你的领域中。
- 如何将计算思维应用到你的领域中？你可以大体上描写计算思维或者你可以思考不同计算思维概念之间的区别，并描述你将哪一个概念应用到了你的领域中。

5-2-2 项目，第二部分

第二部分：制作一份将计算思维整合进入你的教学的计划。

提示：在项目的第二部分中，你将会记录下你是如何计划将至少一个计算思维的概念整合进入

你的课堂、活动、单元、项目、模型或者课程中。

- 将至少一个计算思维的概念整合入你的课程计划中（分解，模式识别，抽象化，算法开发等等）
 - 课程计划（一份关于学生需要学习的内容和如何将其有效率的在课堂、演讲或者某一时期实现的课程指南）
 - 评价（一个活动、测试或者一个项目可以用来衡量或评测学生是否完成了学习目标和达到的程度）
 - 课程活动（一个或一组活动用来给学生完成以支持课堂目标）
 - 项目（一项学生可以通过在特定时间求解问题，调查或回答复杂的挑战、问题或者任务的工作）
 - 单元计划（或者模块规划；一组可以被用来分享共同目标的互相关联的课程、活动、评价以及项目，通常可以在数天或者数周内完成）。
- 一个可以用来向你的学生、家长、或者学校社区来解释以下内容的交流计划：
 - 什么是计算思维
 - 为什么计算思维很重要
 - 你会如何以及你会在哪里开始将计算思维整合入你的教学（例如：科目、学期、课程、项目、活动、课后学习、服务项目、模组等等）
- 其他

5-3 样例项目

- [整合计算思维样例课程计划](#)：[探索计算思维课程计划](#)（或者参考附件一）
- [交流计划案例](#)（或者参考附件四）
- [策略计划案例](#)（或者参考附件五）

注意：如果你使用了其中的一个案例，请留下一份拷贝并编辑它以使其跟你自身情况相适应。

1. 你在教学实践中使用了下列哪些计算思维的概念？选择一个或多个

- ☐ 抽象化和模式推广
- ☐ 算法开发
- ☐ 分解

- ☐ 模式识别
- ☐ 其他

2. 你会创建什么类型的计划？选择一个或多个

- ☐ 策略
- ☐ 课程
- ☐ 活动
- ☐ 单元或模块
- ☐ 项目
- ☐ 和其他科目的老师一起进行的联合项目或课程
- ☐ 跟学生、老师、管理者或家长的交流计划
- ☐ 其他

3. 你打算做些什么？

描述并联系策略计划，或者给课堂、活动或者学生项目的计划。如何将计算思维嵌入其中？

你的时间框架是怎样的？你何时实行这个计划？你的项目会持续多久？

4. 你如何知道项目是否成功了？

建立了跨课程的联系，提升了学生成绩，提升了学生的参与度，更加有效率的课程，丰富了讨论和观点等等。

5-4 总结

恭喜你完成了本课程！

在完成本课程后，我们希望你对什么是计算思维以及如何将其应用到各个学科中有了更进一步的理解。应用计算思维过程给学生提供了一个亲自动手来探究和研究课题的机会。简单的说，计算思维的步骤如下：

- 分解： 将数据、过程、或者问题分解为较小的可管理或可解决的部分。

- 模式识别：观察模式、趋势以及数据中的规律。
- 抽象化：确认生成模式的一般规律。
- 算法开发：开发一系列分步指示，用以解决一些相似的问题。

这里有一些额外的资源可以让你继续学习计算思维以及学习如何将其整合入你的课程之中。

- 访问 g.co/exploringCT（或者参考附件一），你可以获得我们收藏的计算思维整合资源，包括课堂准备课程、演示、程序和视频。
- 你可以在世界各地访问 [course community](#)（或者参考附件二）来于其他教育者和相关领域的专家一起继续讨论计算思维的话题。

本课程英文网址：

https://computationalthinkingcourse.withgoogle.com/course?use_last_location=true

感谢翻译组：

毛澄洁 北京景山学校高级教师

项华 北京师范大学物理学系教授

梁志成 广州市执信中学高级教师

覃芳 北京景山学校高级教师

李忻玮 美国哥伦比亚大学，计算机数据科学专业在读研究生

睦衍波 北京景山学校高级教师

李雅楠 北京景山学校一级教师

黎小说 盒子鱼英语 CEO

张瑞林 北京汇文学校一级教师

项杰庭 北京数字创客教育科技研究院工程师