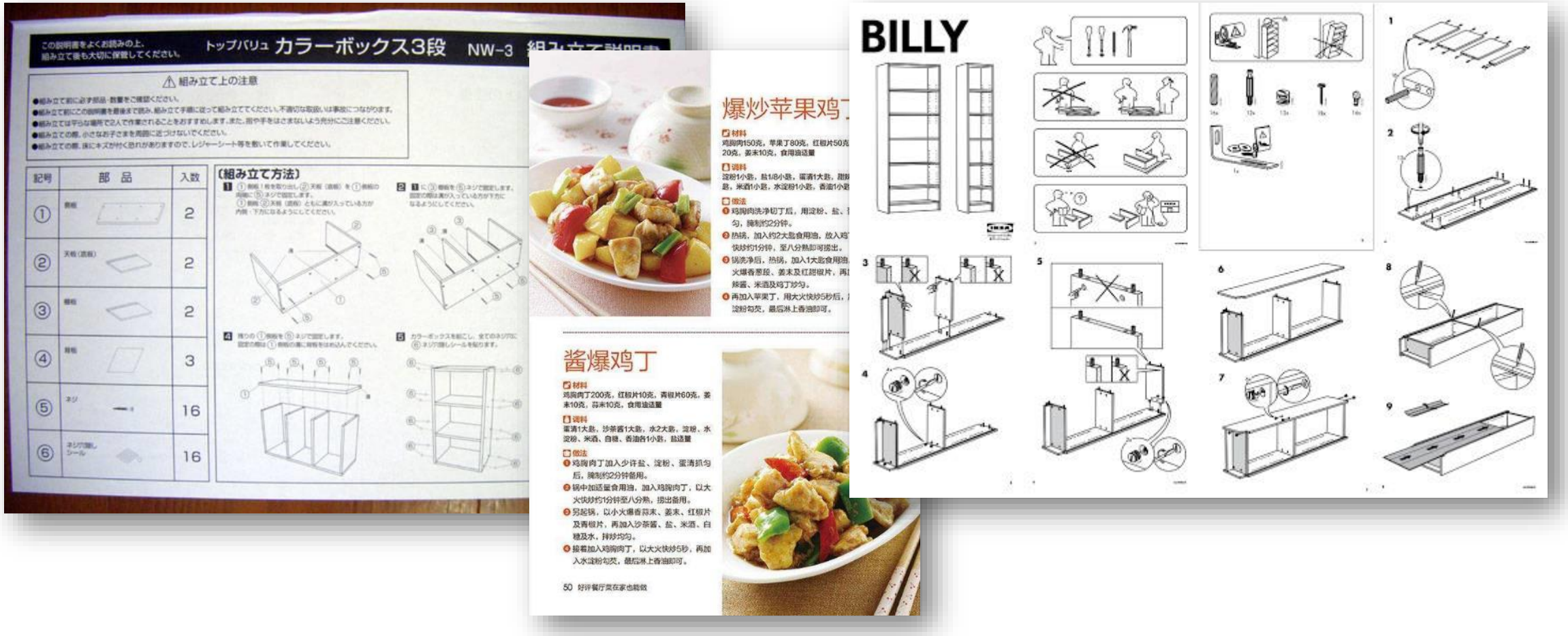
数据科学导论

Introduction to Data Science

第4章
数据的计算与程序表达



- 解决一个问题的进程
- 可机械执行的一系列步骤精准而明确的规范

什么是算法

In mathematics and computer science, an algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation.

Algorithm

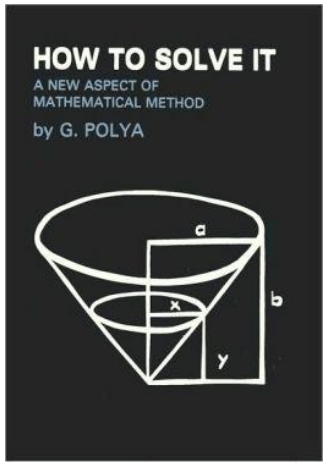
- Input
- Output
- Definiteness
- Effectiveness
- Finiteness

什么是算法

- 正如大名鼎鼎的Polya所说，为的是在遇到问题时，我们知道“How to solve it!”
- 对于每一个算法都有这样的一个过程：

  设计 → 证明 → 应用

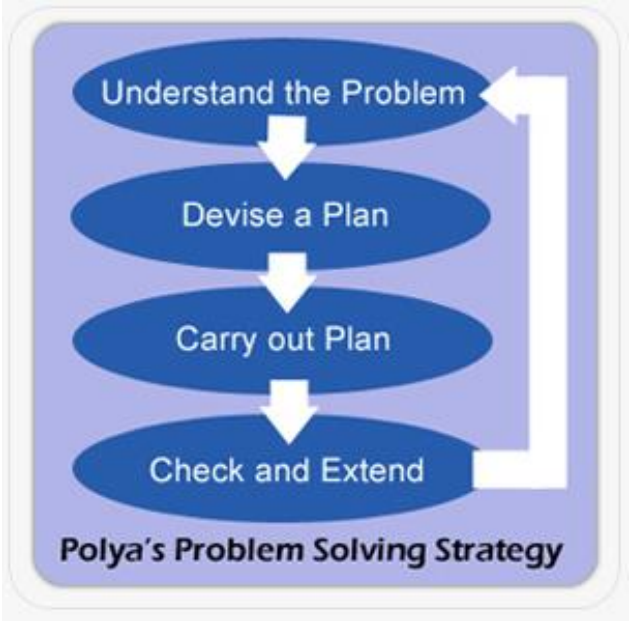
- 而我们学习算法其实也是对这三个方面有着不同的侧重。

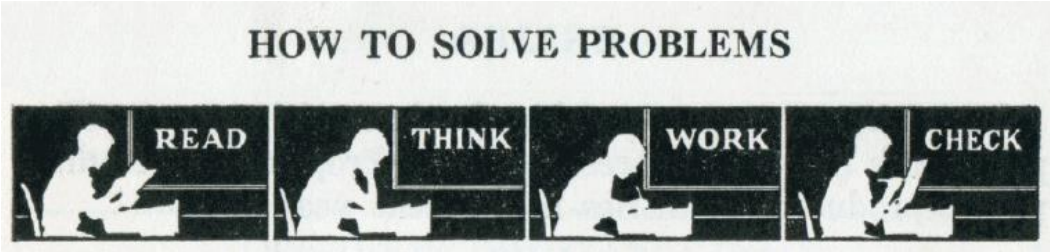


算法的作用

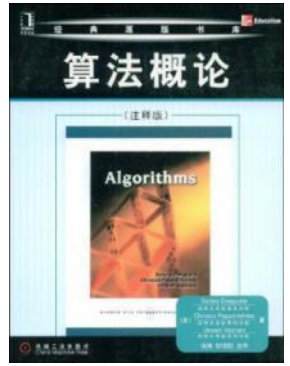
Polya's Problem Solving Steps

1. Understand the problem.
2. Devise a plan for solving the problem.
3. Carry out the plan.
4. Evaluate the solution for accuracy and its potential as a tool for solving other problems.





算法的作用



给程序设计带来的启示

1. Understand the problem.
2. Get an idea of how an algorithmic procedure might solve the problem.
3. Formulate the **algorithm** and represent it as a program.
4. Evaluate the program for accuracy and for its potential as a tool for solving other problems.

Problem → Solution → Algorithm Design → Mathematics Model →Application

算法的作用

算法与程序

- 图灵奖获得主瑞士科学家Niklaus Wirth，Pascal语言的发明者和结构化程序设计创始者，在1976年出版了《算法 + 数据结构 = 程序设计》，提出了著名的公式：

$$Program = DS + Algorithm$$

  - 算法：解决问题的计算方法（步骤）
  - 数据结构：数据存储的模型
  - 语言：描述算法和数据结构的工具
  - 程序：用语言描述出来的算法和数据结构

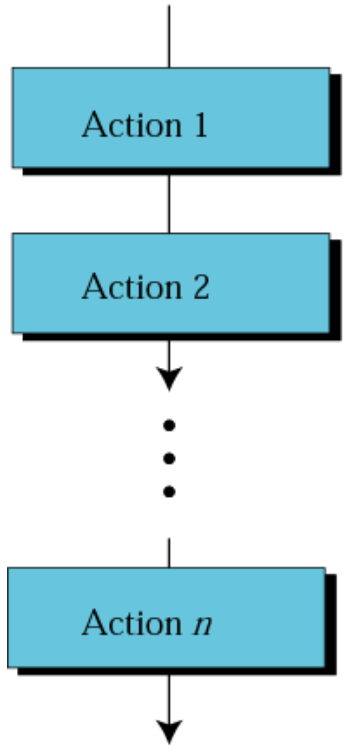
算法的作用

- 算法在中国古代文献中称为“术”，最早出现在《周髀算经》、《九章算术》。特别是《九章算术》，给出四则运算、最大公约数、最小公倍数、开平方根、开立方根、求素数的埃拉托斯特尼筛法，线性方程组求解的算法。三国时代的刘徽给出求圆周率的算法：刘徽割圆术
- 自唐代以来，历代更有许多专门论述“算法”的专著

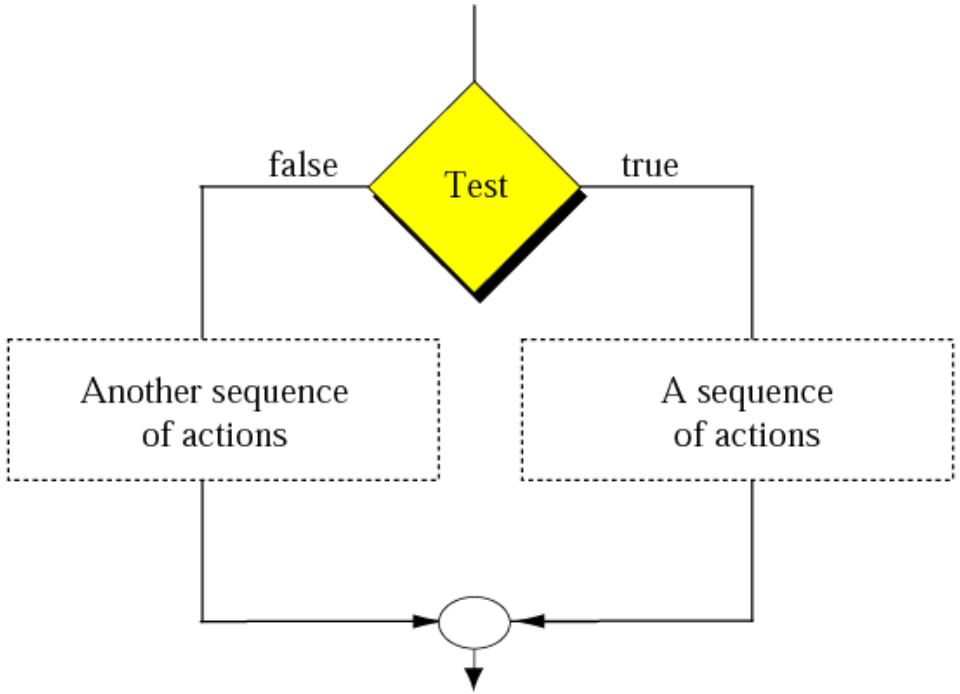
算法的历史

- 而英文名称”algorithm”来自于9世纪波斯数学家花拉子米，因为比阿勒·霍瓦里松在数学上提出了算法这个概念。“算法”原为“algorism”，即“al-Khwarizmi”的音转，意思是“花拉子米”的运算法则，在18世纪演变为“algorithm”。
- 欧几里得算法被人们认为是史上第一个算法。
- 第一次编写程序是Ada Byron于1842年为巴贝奇分析机编写求解解伯努利微分方程的程序，因此爱达·勒芙蕾丝被大多数人认为是世界上第一位程序员。
- 因为“well-defined procedure”缺少数学上精确的定义，19世纪和20世纪早期的数学家、逻辑学家在定义算法上出现了困难。20世纪的英国数学家图灵提出了著名的图灵论题，并提出一种假想的计算机的抽象模型，这个模型被称为图灵机。图灵机的出现解决了算法定义的难题，图灵的思想对算法的发展起到了重要的作用。

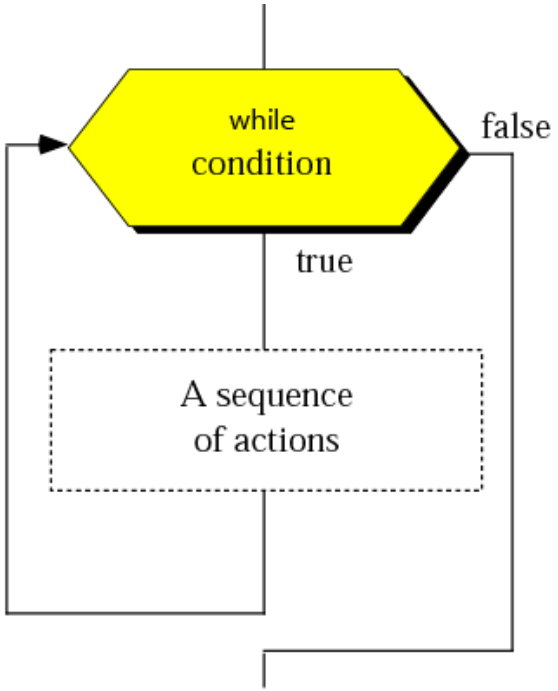
算法的历史



a. Sequence

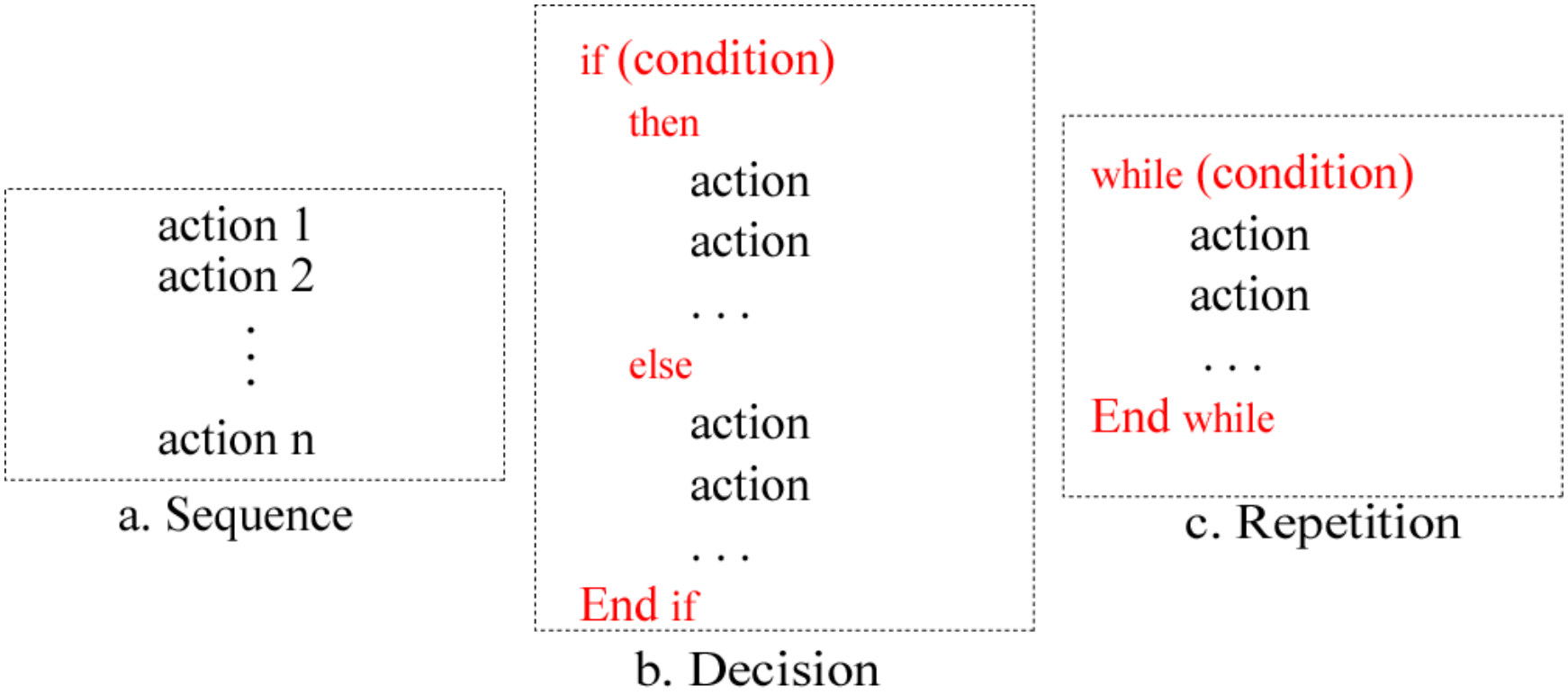


b. Decision

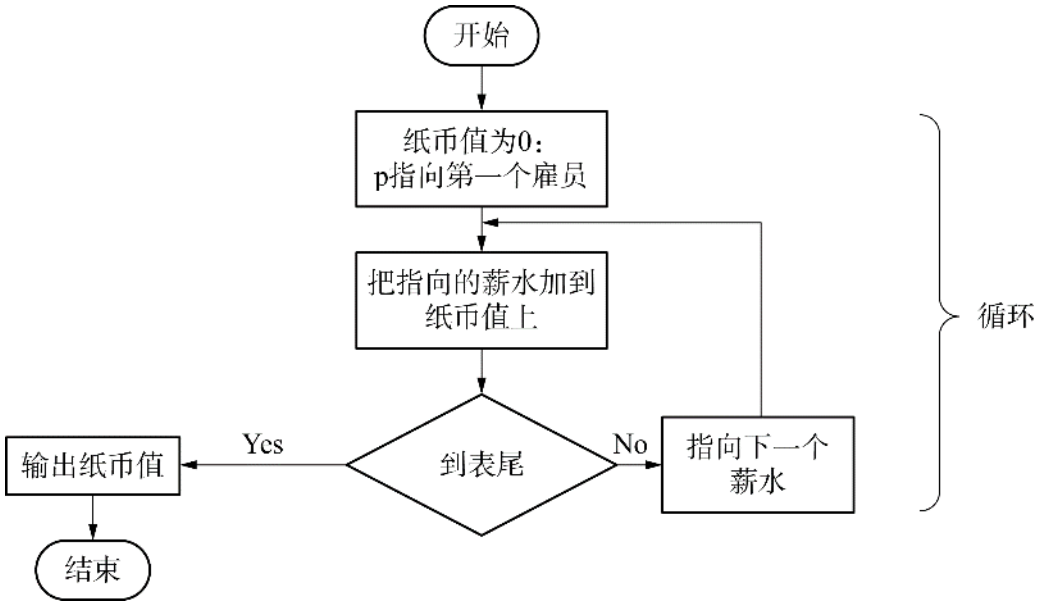


c. Repetition

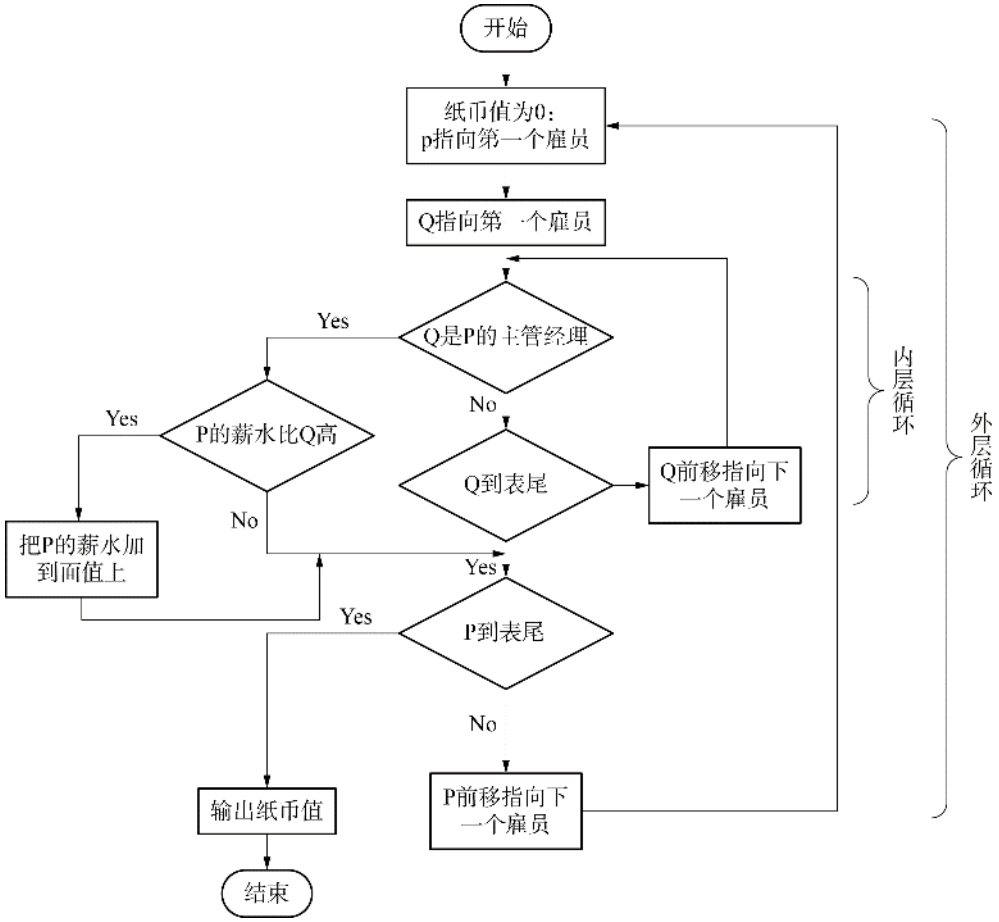
算法的控制结构与流程图



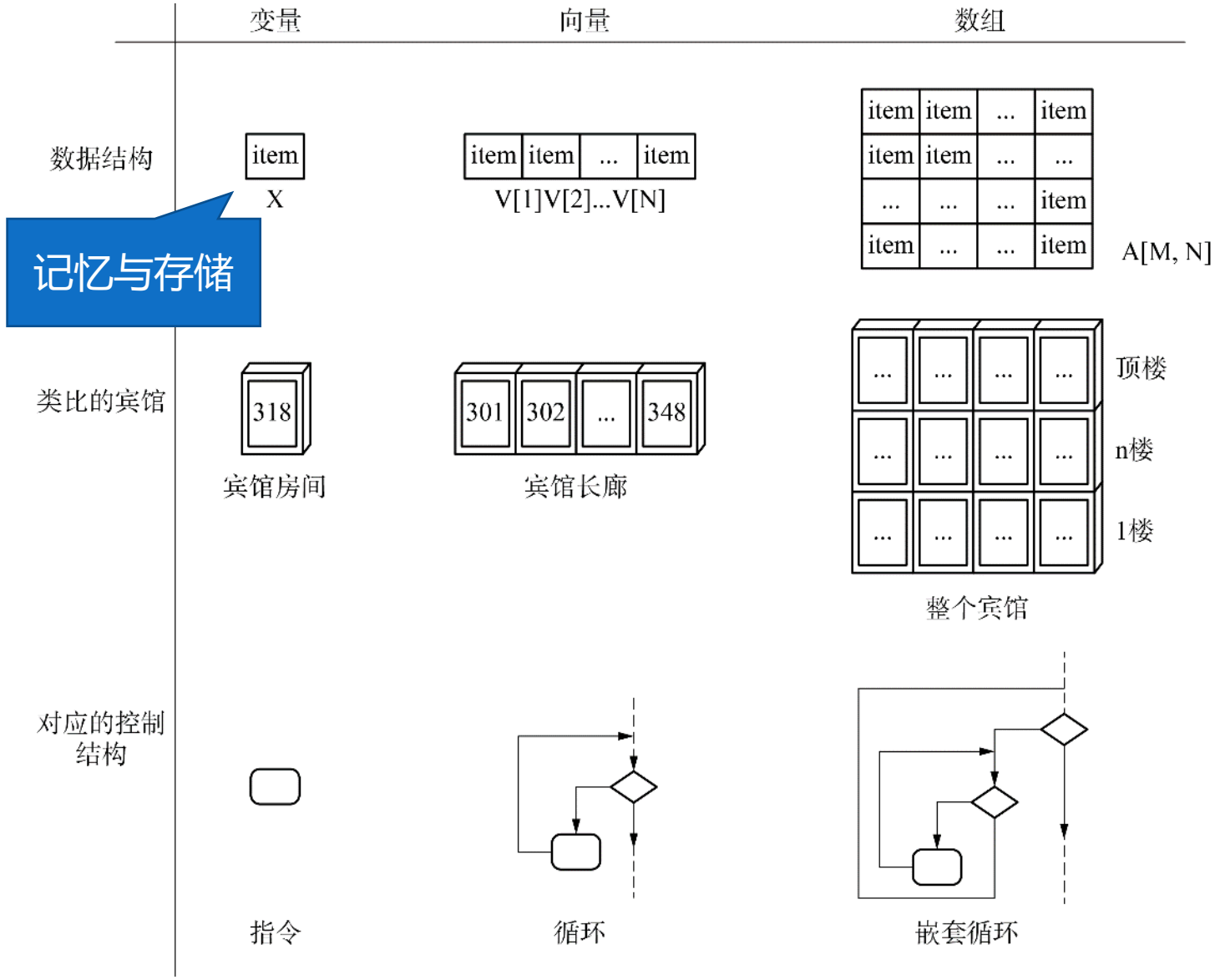
算法的控制结构与流程图

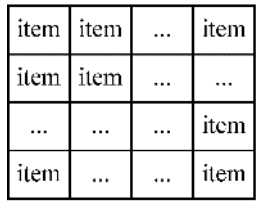


三种控制结构组合运用

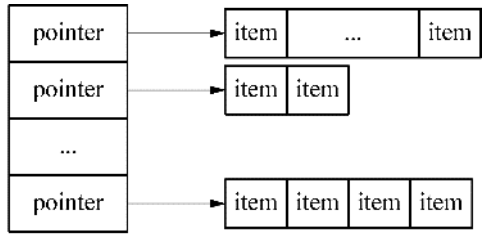


算法的控制结构与流程图





数组



向量中的向量

算法中的变量、向量和数组

第4章
数据的计算与程序表达

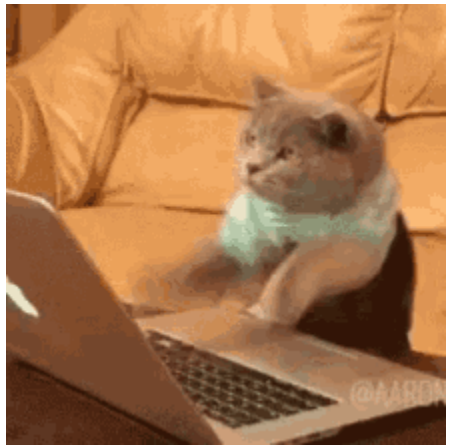
1 数据的计算

2 **算法分析**

3 数据结构与算法的关系

4 算法实例

5 计算机编程语言



好的算法

劣质算法

鱼和熊掌
不可兼得

评价一个算法

- 正确执行
- 执行算法所耗费时间
- 执行算法所耗费存储空间
- 易于理解、可读、编码、调试
- 健壮性（鲁棒性）

算法评价

- 算法的时间复杂度是一个函数，它定性描述该算法的运行时间。
- 时间复杂度常用大O符号表述，不包括这个函数的低阶项和首项系数。

查找 25 ： 1 次

查找 -85: n 次

查找第n/2个值：n/2 次

线性表顺序查找的时间复杂度
$O(n)$

时间复杂度

```
int num1,num2;
for (int i=0;i<n;i++)
{
    num1 += 1;
    for (int j=1;j<=n;j*=2)
    {
        num2 += num1;
    }
}
```

1

n

$n \times \log_2(n)$

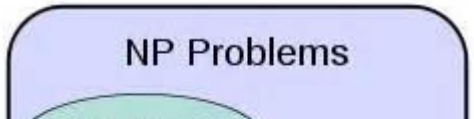
$$f(n) = 4 + 3n + 2n\log_2 n$$

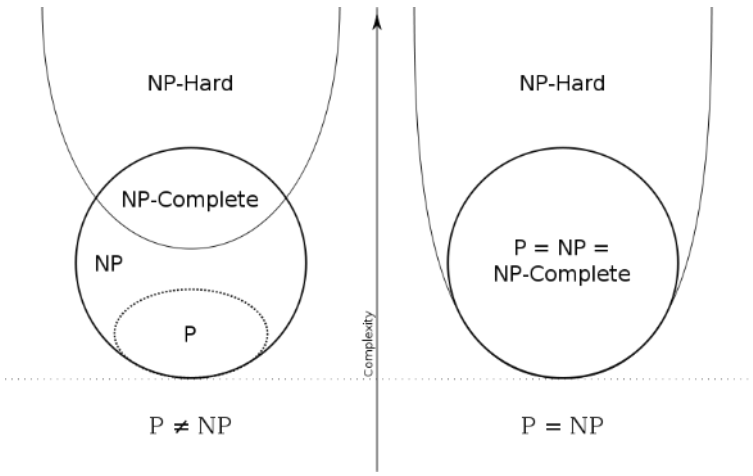
$$O(n\log_2 n)$$

时间复杂度

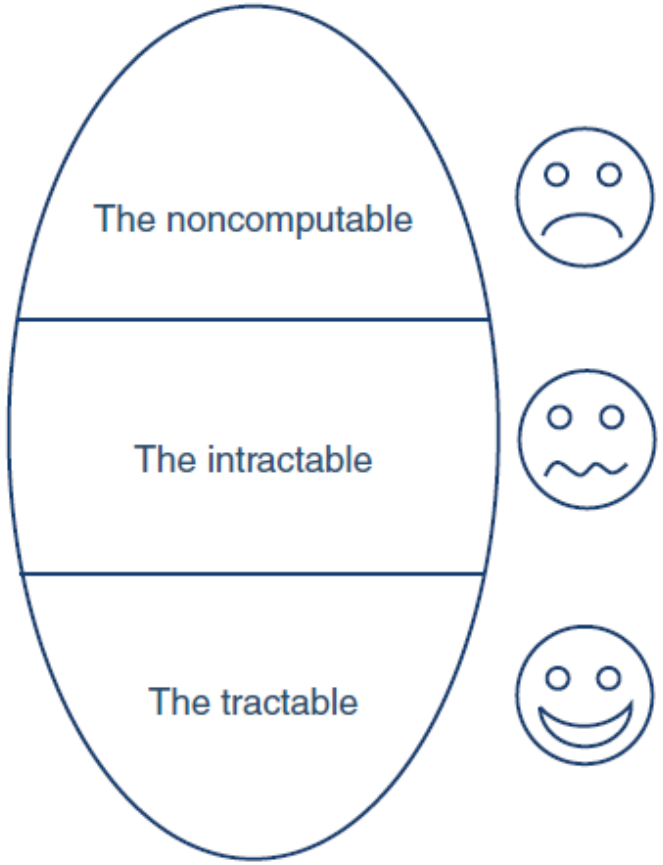

无法容忍
n! 2ⁿ n²
n log₂n
n
N
n
√n
log₂n
1
好
推荐

P = NP ?


NP Problems
P Problems
NP Complete


NP-Hard
NP-Complete
NP
P
Complexity
P ≠ NP
NP-Hard
P = NP =
NP-Complete
P = NP

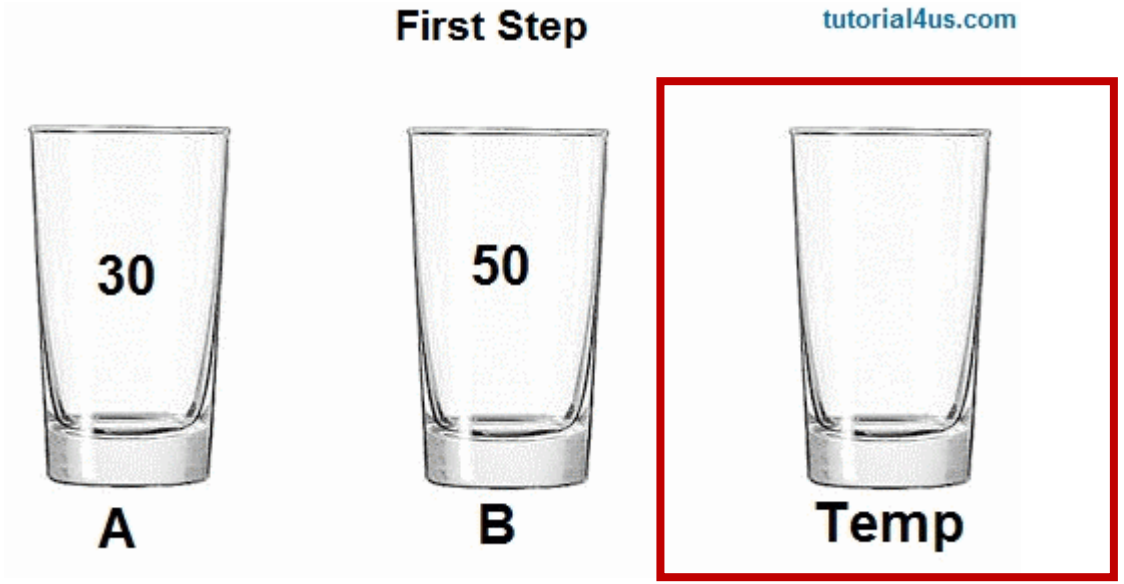
时间复杂度



- Noncomputable problems have no algorithmic solution;
- Algorithms for intractable problems do exist but only with exponential or higher order of complexity;
- Tractable problems can be solved with polynomial time algorithms.

时间复杂度

空间复杂度是对一个算法在运行过程中临时占用存储空间大小的量度



额外所需空间
O(1)

单位存储空间的价格越来越低
人们更关注运行效率

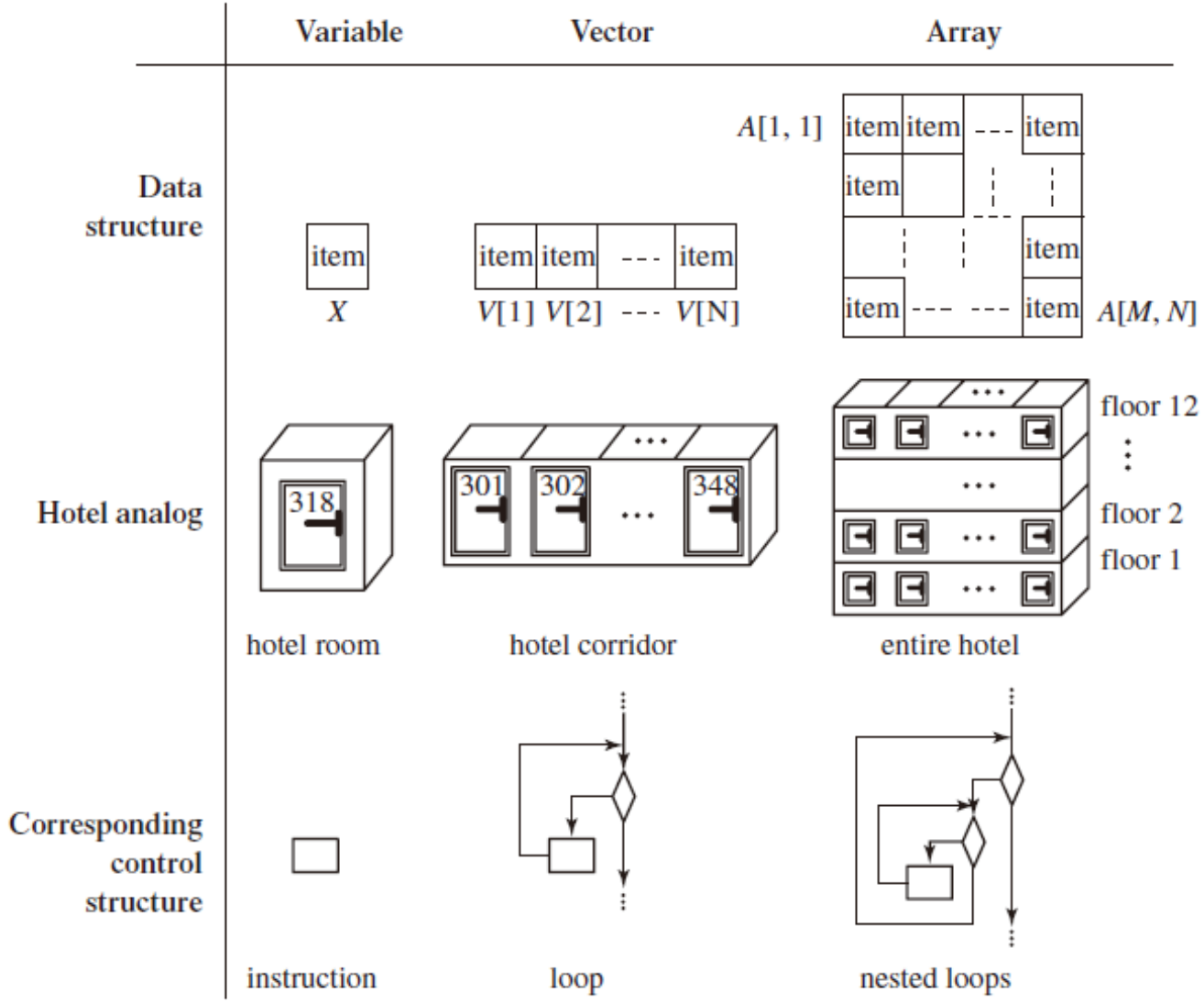
空间复杂度

第4章
数据的计算与程序表达

- 数据结构是用来干什么的？装数据的，使得算法能在上面做有效的计算。
  - 数据是信息之源泉，没有数据就没有信息，没有大量数据就没有有效信息。
  - 而算法是把信息从数据中提取出来的手段，没有算法的提取，数据还是数据，永远不会变成有用的信息。

- 数据结构和算法的关系应该是相辅相成，融为一体的。

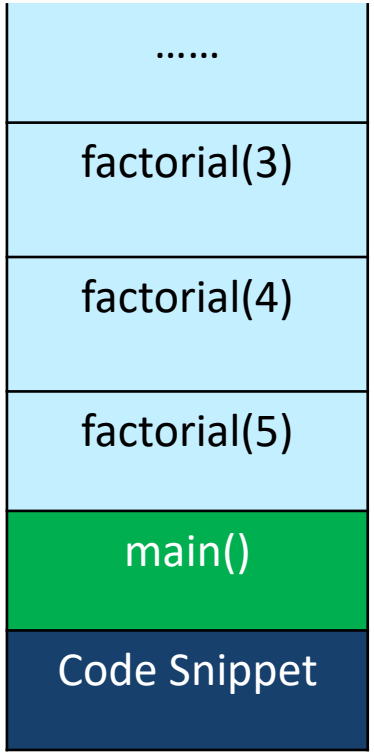
程序 = 数据结构 + 算法




```
for (i=0;i<n;++i)
{
    for (j=0;j<n;++j)
    {
        cout<<a[i][j];
    }
}
```

数组（向量）与循环

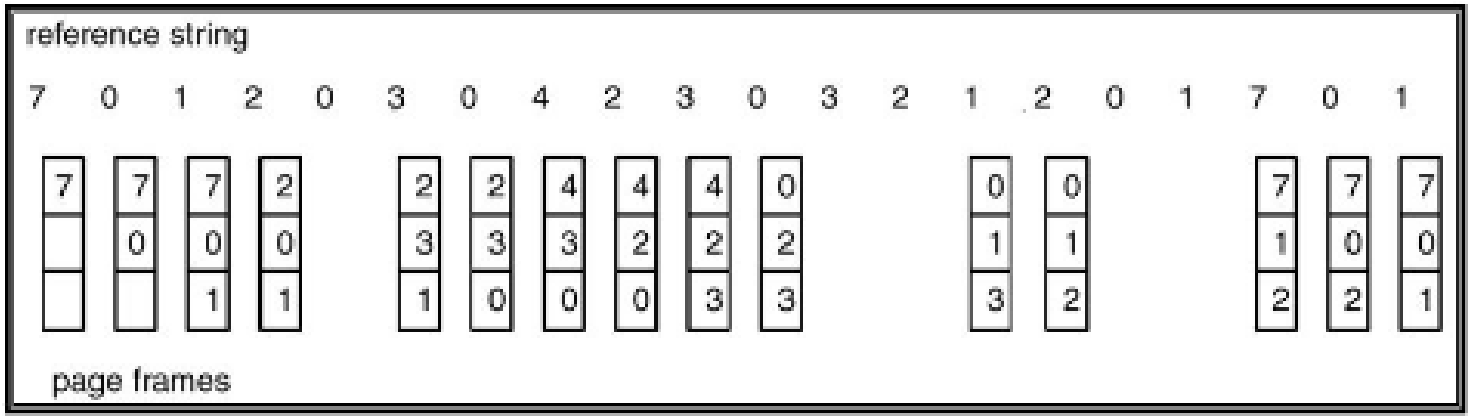
```
int factorial(int n)
{
    if (n==1)
        return 1;
    else
        return n*factorial(n-1);
}
int main()
{
    int a = 5;
    int b = factorial(a);
    return 0;
}
```



函数调用与堆栈

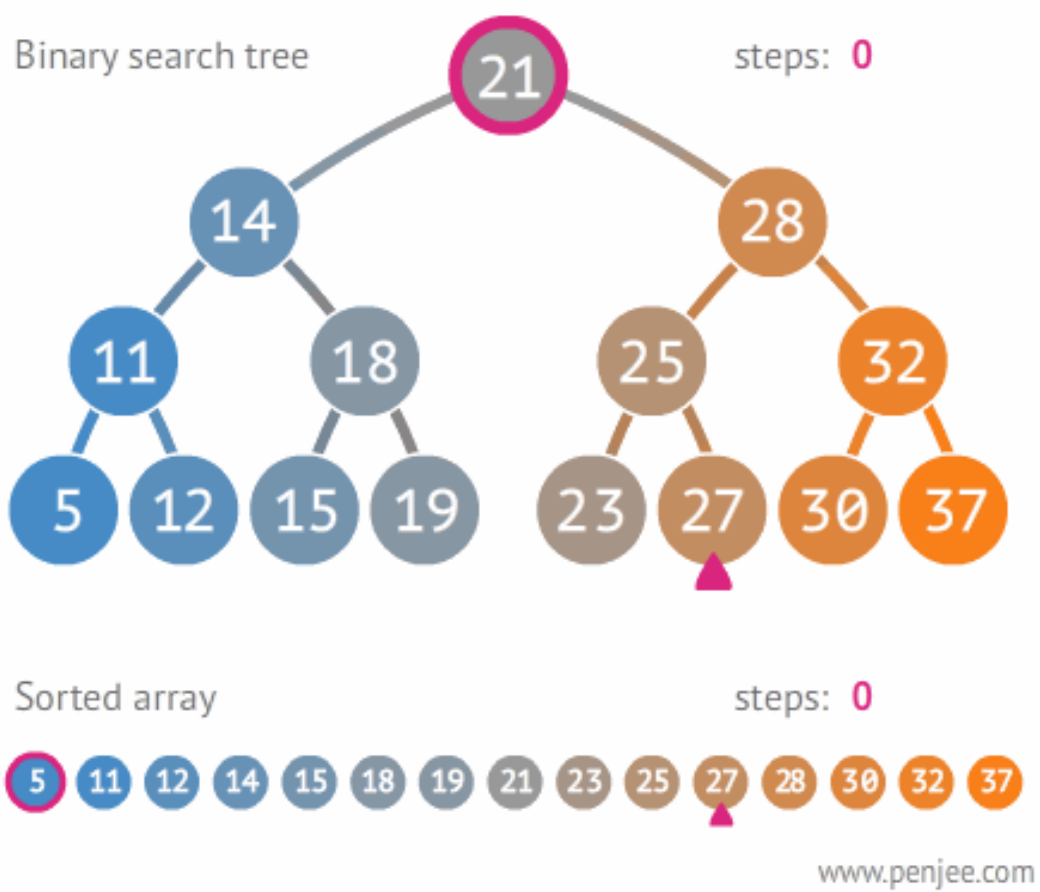
递归与堆栈

FIFO Page Replacement



FIFO页面置换算法

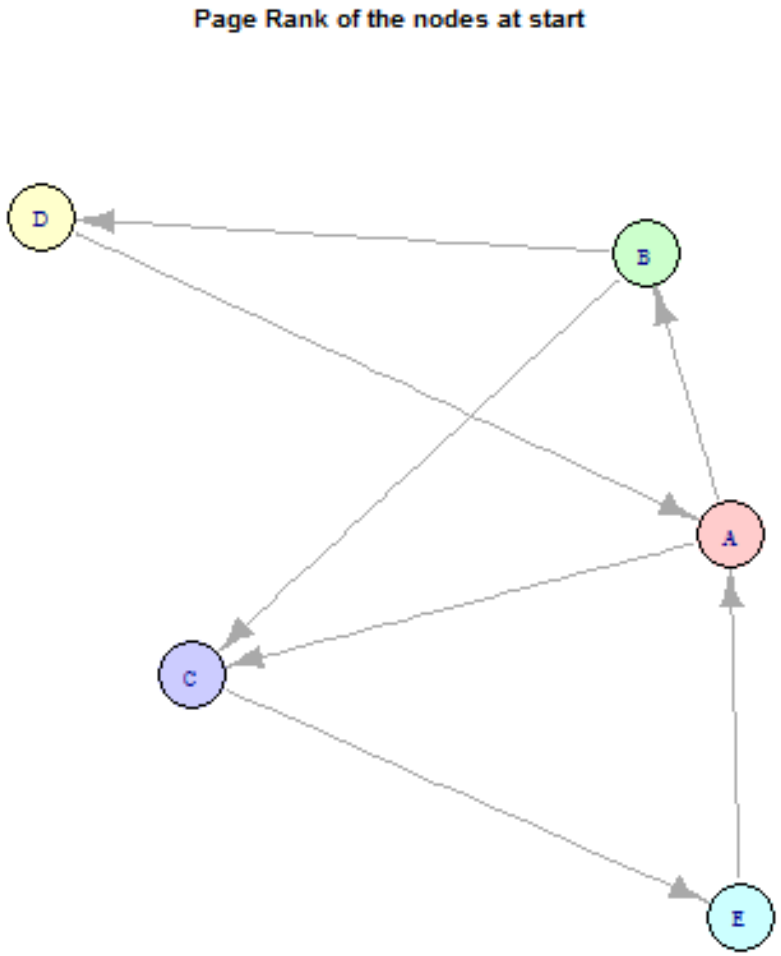
页面置换与队列

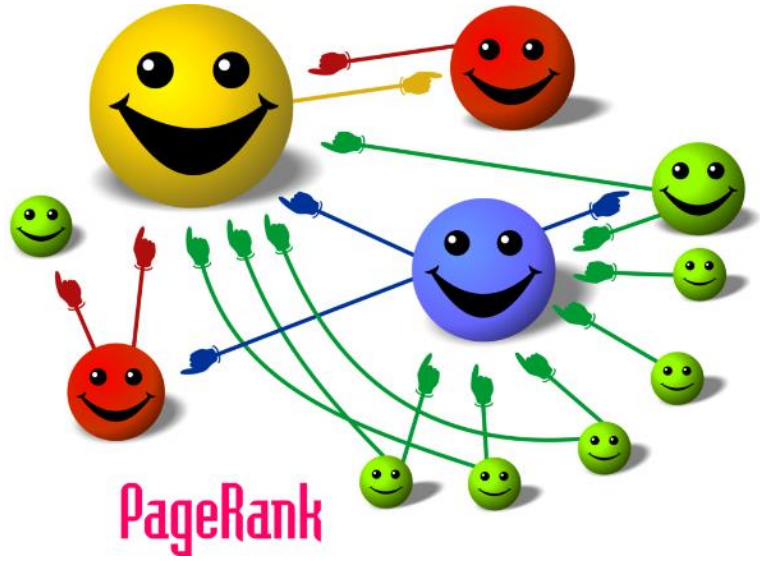

Binary search tree
steps: 0
21
14
28
11
18
25
32
5
12
15
19
23
27
30
37
Sorted array
steps: 0
5 11 12 14 15 18 19 21 23 25 27 28 30 32 37
www.penjee.com

二叉搜索树

排序与树

	Rank
A	0.2
B	0.2
C	0.2
D	0.2
E	0.2





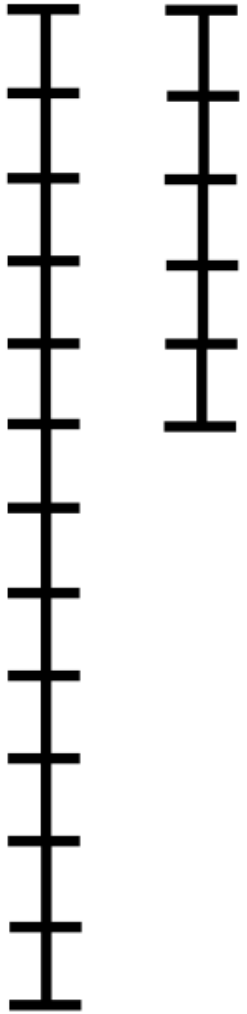
PageRank，又称网页排名是Google公司所使用的对其搜索引擎搜索结果中的网页进行排名的一种算法。

PageRank与图

第4章
数据的计算与程序表达

在数学中，辗转相除法，又称欧几里得算法（英语：Euclidean algorithm），是求最大公约数的算法。辗转相除法首次出现于欧几里得的《几何原本》（第VII卷，命题i和ii）中，而在中国则可以追溯至东汉出现的《九章算术》。

```
输入：自然数 a、b
输出：a、b 的最大公约数
步骤：
    第 1 步：令 r 为 a / b 所得余数
    第 2 步：若 r = 0，则算法结束，b 即为答案；否则置 a←b，b←r，转到第 1 步。
```



两条线段长分别可表示252和105
则其中每一小分段长代表最大公约数21。只要辗转地从大数中减去小数
直到其中一段的长度为0
此时剩下的一条线段的长度就是最大公因数。

欧几里德算法

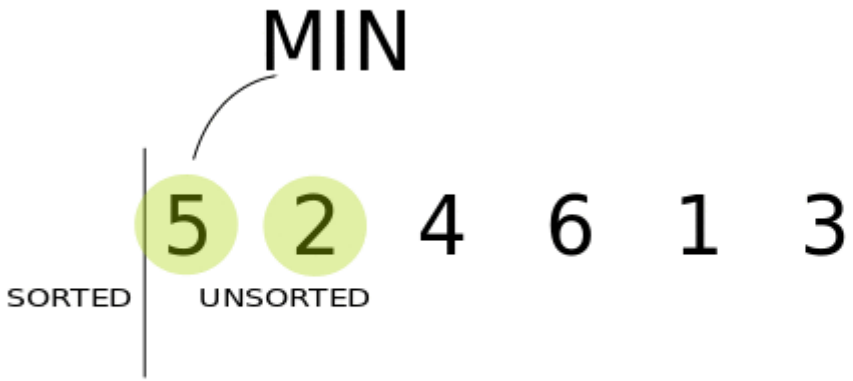
- 排序（Sort）是指对一些数据的重新组织，使得数据由大到小（降序）或者由小到大（升序）存储。排序是很一种基本的要求。
- 我们所感兴趣的是如何寻找到一个好（计算速度快或占用内存少）的办法来进行排序。
  - 插入排序
  - 选择排序

排序算法

插入排序（Insertion Sort）是一种简单直观的排序算法。
它的工作原理是通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。

排序算法——插入排序

选择排序（Selection sort）是一种简单直观的排序算法。它的工作原理如下：
首先在未排序序列中找到最小元素，存放到排序序列的起始位置；
然后再从剩余未排序元素中继续寻找最小元素，然后放到已排序序列的末尾；
以此类推，直到所有元素均排序完毕。



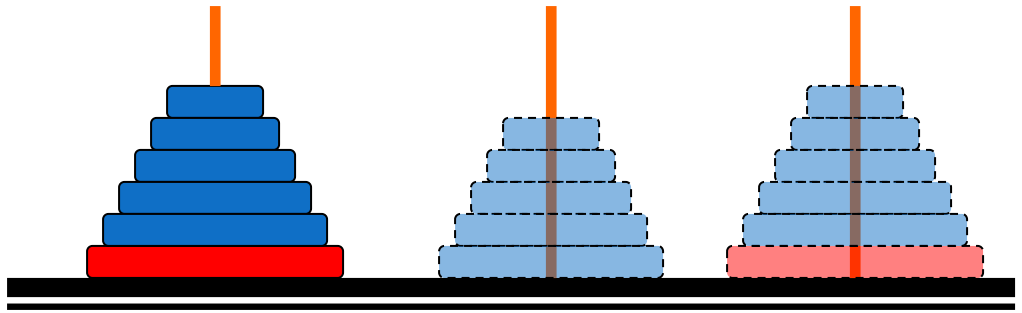
排序算法——选择排序

4.4 算法实例

名称	数据对象	稳定性	时间复杂度		额外空间复杂度	描述
			平均	最坏		
冒泡排序	数组	✓	$O(n^2)$		$O(1)$	（无序区，有序区）。 从无序区透过交换找出最大元素放到有序区前端。
选择排序	数组	✗	$O(n^2)$		$O(1)$	（有序区，无序区）。 在无序区里找一个最小的元素跟在有序区的后面。对数组：比较得多，换得少。
	链表	✓				
插入排序	数组、链表	✓	$O(n^2)$		$O(1)$	（有序区，无序区）。 把无序区的第一个元素插入到有序区的合适的位置。对数组：比较得少，换得多。
堆排序	数组	✗	$O(n \log n)$		$O(1)$	（最大堆，有序区）。 从堆顶把根卸出来放在有序区之前，再恢复堆。
归并排序	数组	✓	$O(n \log^2 n)$		$O(1)$	把数据分为两段，从两段中逐个选最小的元素移入新数据段的末尾。 可从上到下或从下到上进行。
			$O(n \log n)$		$O(n) + O(\log n)$ 如果不是从下到上	
	链表				$O(1)$	
快速排序	数组	✗	$O(n \log n)$	$O(n^2)$	$O(\log n)$	（小数，基准元素，大数）。 在区间中随机挑选一个元素作基准，将小于基准的元素放在基准之前，大于基准的元素放在基准之后，再分别对小数区与大数区进行排序。
	链表	✓				
希尔排序	数组	✗	$O(n \log^2 n)$	$O(n^2)$	$O(1)$	每一轮按照事先决定的间隔进行插入排序，间隔会依次缩小，最后一次一定要是1。
计数排序	数组、链表	✓	$O(n + m)$		$O(n + m)$	统计小于等于该元素值的元素的个数i，于是该元素就放在目标数组的索引i位（i≥0）。
桶排序	数组、链表	✓	$O(n)$		$O(m)$	将值为i的元素放入i号桶，最后依次把桶里的元素倒出来。
基数排序	数组、链表	✓	$O(k \times n)$	$O(n^2)$		一种多关键字的排序算法，可用桶排序实现。

- 均按从小到大排列
- k代表数值中的"数位"个数
- n代表数据规模
- m代表数据的最大值减最小值

- How many moves are need to move all the disks to the third peg by moving only one at a time and never placing a disk on top of a smaller one.



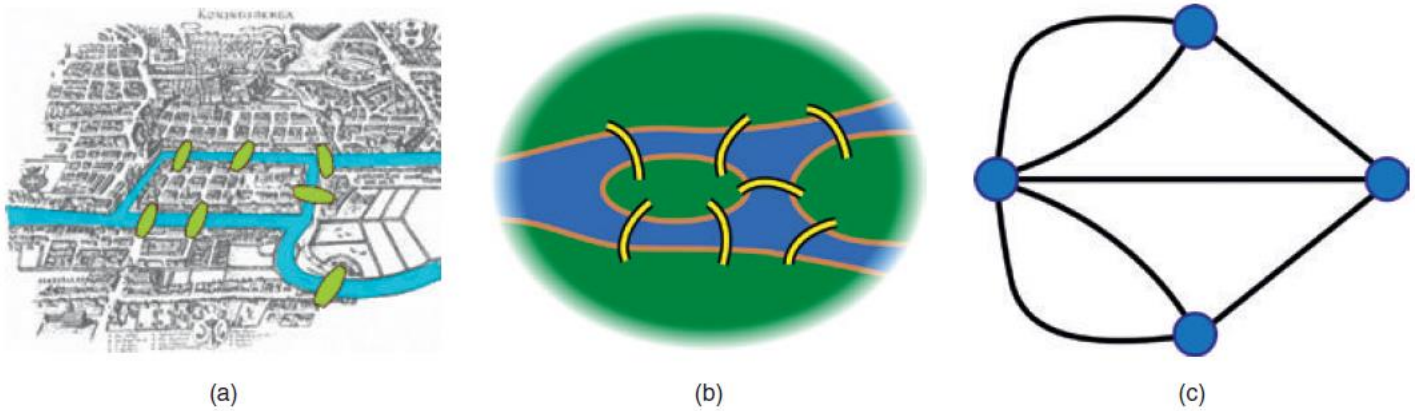
```
Hanoi(n, A, C)
        Hanoi(n-1, A, B);
        Move(A, C);
        Hanoi(n-1, B, C);
end
```

汉诺塔问题

柯尼斯堡七桥问题（Seven Bridges of Königsberg）是图论中的著名问题。

这个问题是基于一个现实生活中的事例：当时东普鲁士柯尼斯堡（今日俄罗斯加里宁格勒）市区跨普列戈利亚河两岸，河中心有两个小岛。

小岛与河的两岸有七条桥连接。在所有桥都只能走一遍的前提下，如何才能把这个地方所有的桥都走遍？

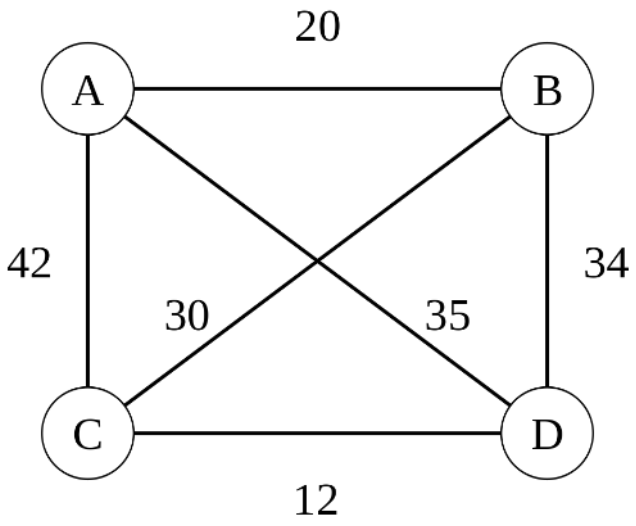


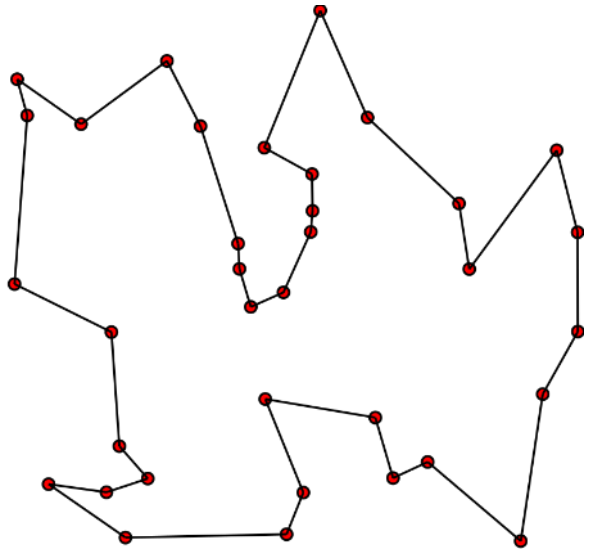
七桥问题

旅行商问题（最短路径问题，TSP）是这样一个问题：

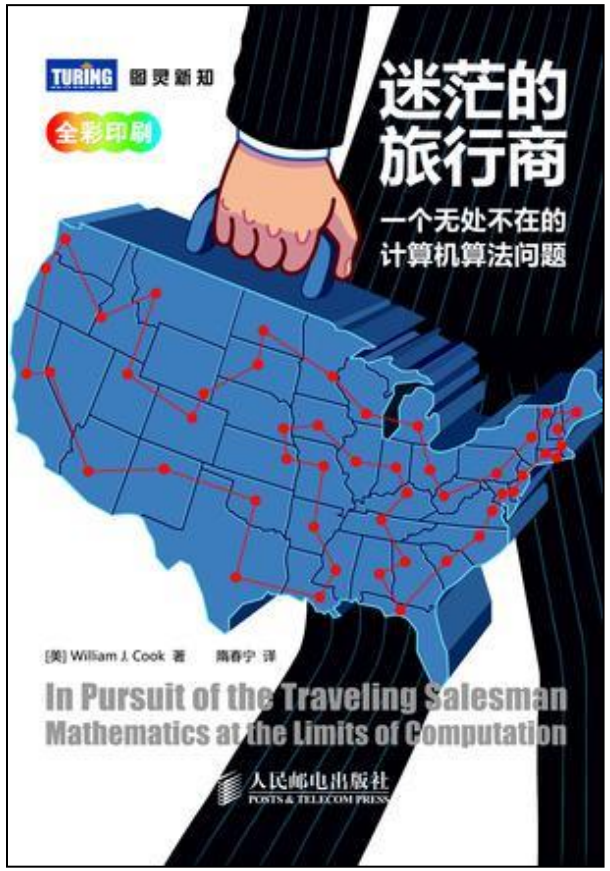
给定一系列城市和每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路。

它是组合优化中的一个NP困难问题，在运筹学和理论计算机科学中非常重要。

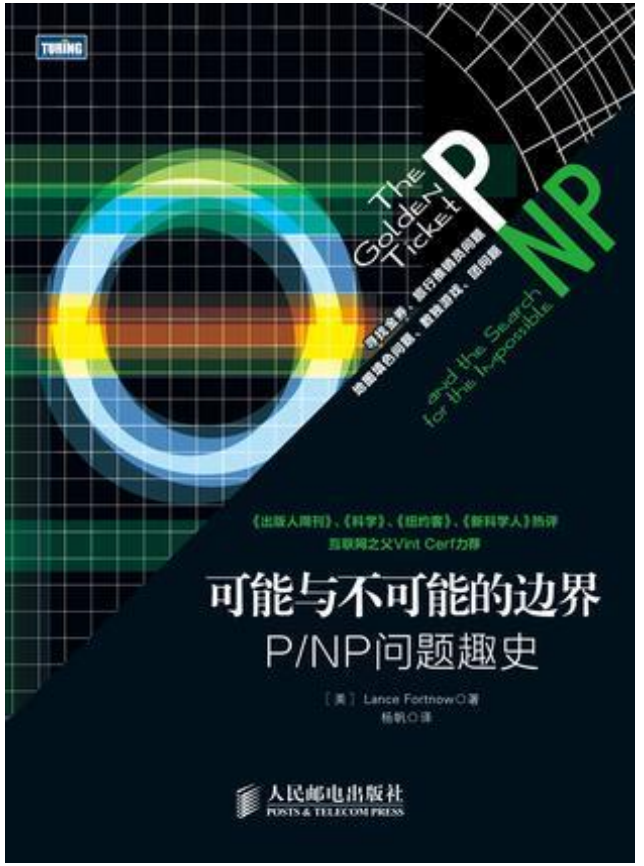




旅行商问题

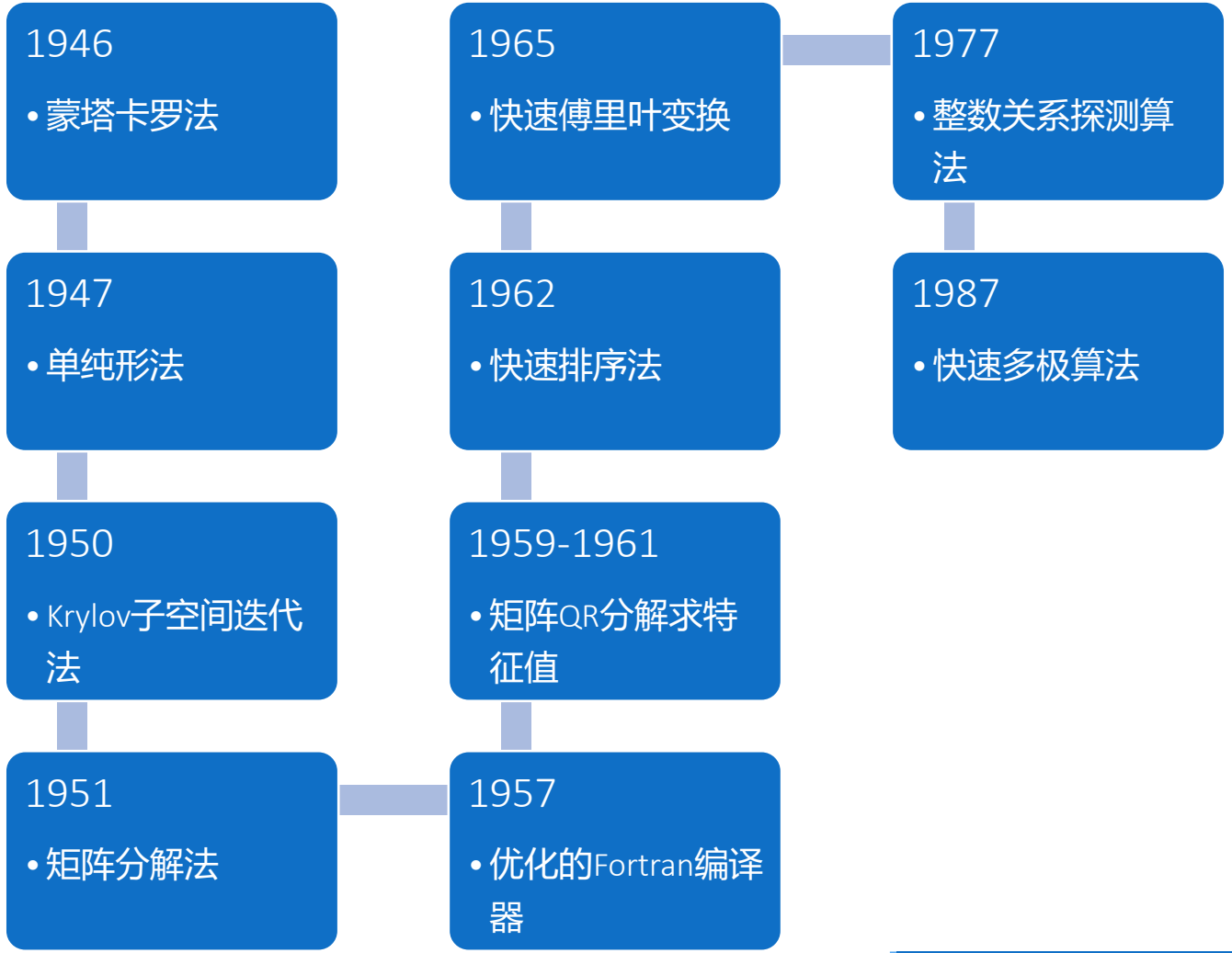


关于经典的TSP问题的一切。TSP问题看似简单，特别是在问题规模较小时，最优解似乎是不言自明的，但当问题规模不断扩大，即使是人脑这样的“超大规模并行”系统也会立刻感到无所适从、进而“迷茫”。



P指的是用计算机能很快求解的问题，NP指的则是我们想找到最优解的问题。如果P=NP，那么我们将很容易找到任意给定问题的解。这意味着我们所了解的社会将发生剧变，医学、科学、娱乐和人类社会一切任务的自动化程度都将立即发生质的飞跃。然而直到目前我们对P是否等于NP这个问题，一直没有一个明确的答案。

旅行商问题


1946
• 蒙塔卡罗法
1947
• 单纯形法
1950
• Krylov子空间迭代法
1951
• 矩阵分解法
1957
• 优化的Fortran编译器
1959-1961
• 矩阵QR分解求特征值
1962
• 快速排序法
1965
• 快速傅里叶变换
1977
• 整数关系探测算法
1987
• 快速多极算法

20世纪十大算法

搜索引擎

- 索引
- PageRank

密码学

- 公钥加密
- 数字签名

存储

- 纠错码
- 压缩

人工智能

- 模式识别

数据库

- 数据库一致性

最终的极限算法

改变未来的九大算法

- Google的搜索引擎
  - PageRank采用大量增加输入数据量的方法，不仅仅把文字和标题考虑进去，还考虑了大量的“链接”数据
  - 谷歌的强不是强在PageRank算法上，而是赢在数据上，利用了比别人多的数据。
- 算法的“好坏”在没有大量有效数据的支撑下是没有意义的。很多算法的结果的质量完全取决于其和真实数据的拟合程度。
- 正方：Google直言，更多的数据胜过更好的算法。
- 反方：数据只是基础，如何建构起有效的算法、模型比数据本身更重要。

数据对算法的挑战

- 滴滴打车平台使得乘客、出租车司机、车辆信息、交通路况、地理信息等相关数据得到了充分的连接；
- 由于市场的变化和巨头的合并，打车软件背后的算法变得单一，决策权开始集中到少数人手上。
- 加价算法模式，开始成为人们唯一的选择。
- 不公平性的产生！

算法定义一切?

- 算法有意或无意的把人类的成见、误解和偏见等编码到管理我们生活方方面面的软件系统中，进而重现定义我们现在的生活。
  - 能上哪个学校、在哪里工作、能否获得购车贷款、健康保险的缴费标准是多少等各种决策，越来越多地由数学模型和算法决定；
- 现在使用的很多模型和算法都是不透明的，未受到规制的，明明有错却容不得质疑的。

《看门狗 2》

ctOS监控着城市一切

算法的歧视?

第4章
数据的计算与程序表达

1 数据的计算

2 算法分析

3 数据结构与算法的关系

4 算法实例

5 **计算机编程语言**

- 计算机解决问题的过程实质上是机械地执行人们为它编制的指令序列的过程。为了告诉计算机应当执行什么指令，需要使用某种计算机语言。这种计算机语言能够精 确地描述计算过程，称为程序设计语言或编程语言（programming language）。
- 与计算机打交道的理想语言当然是人类用自然语言与计算机（如人机对话）进行对话。遗憾的是，由于自然语言的词语和句子往往有歧义， 既不精确也不简练，至少目前的计算机还不能很好地理解自然语言。
- 所以计算机科学家设计 了人造语言来与计算机进行交流。编程语言是人工设计的形式语言，具有严格的语法和语义， 因此没有歧义的问题。

程序设计语言

机器语言

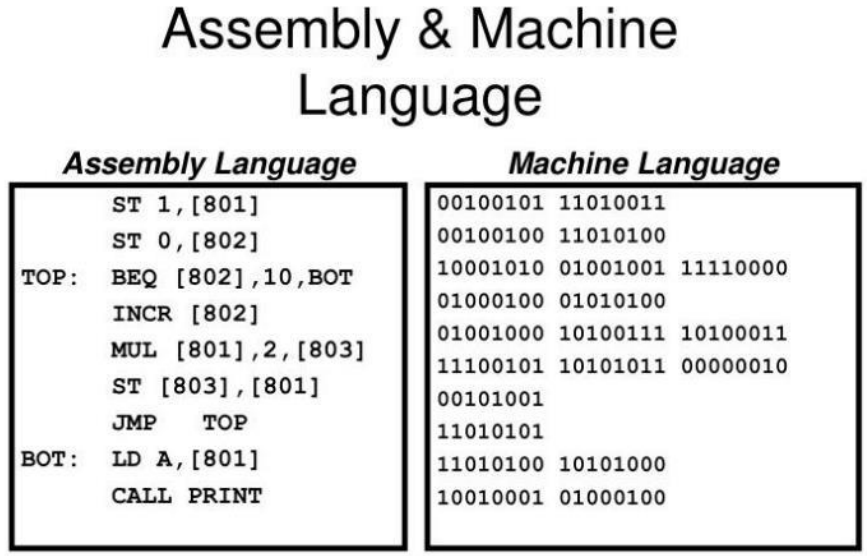
- CPU 制造商在设计某种 CPU 硬件结构的同时，也为其设计了一种“母语”——指令集，这种语言称为机器语言（machine language）。
- 机器语言在形式上是二进制的，即所有指令都是由 0 和 1 组成的二进制序列。利用机器语言写的程序就是二进制指令的序列。

汇编语言

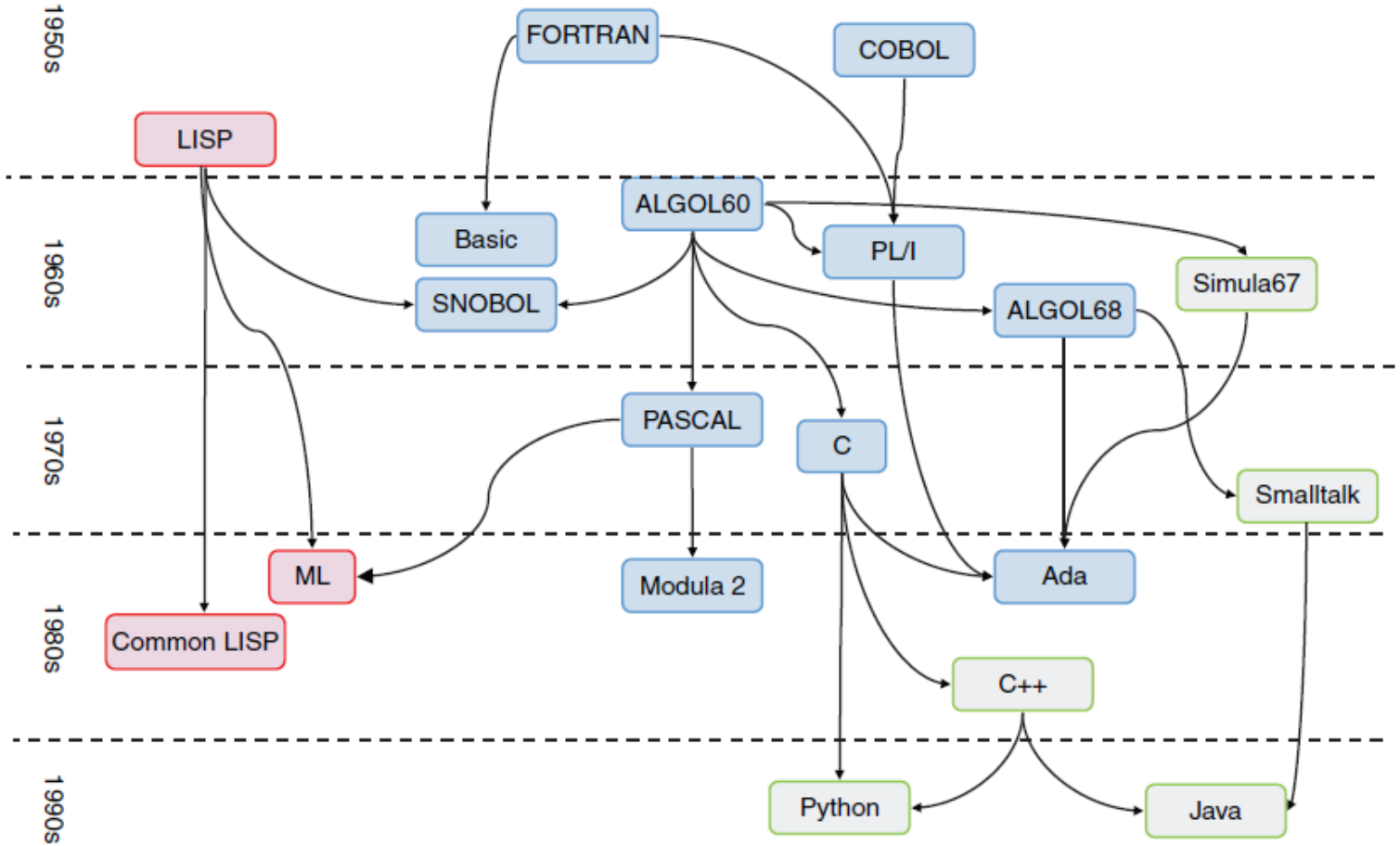
- 为了使编程更容易，人们发明了汇编语言（assembly language）。
- 汇编语言本质上是将机器指令用更加容易为人们所理解和记忆的“助忆符”表现出来。

高级语言

- 高级语言吸收了人们熟悉的自然语言和数学语言的某些成分，因此非常易学、易用、易读；
- 高级语言在构造形式和意义方面具有严格定义，从而避免了语言的歧义性；
- 高级语言与计算机硬件没有关系，用高级语言写的程序可以移植到各种计算机上执行。



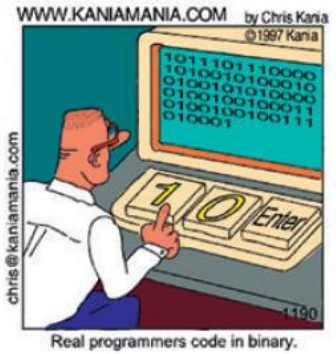
编程语言的类别



Evolutionary graph of programming languages

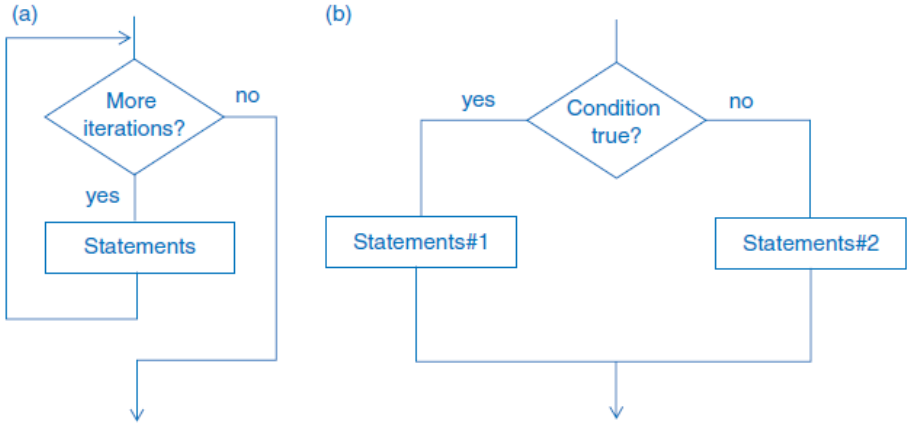
编程语言的发展

- Before the introduction of FORTRAN in the 1950s, programmers had to work in machine code, made up of binary commands , or in assembly languages.



Real programmers code in binary.

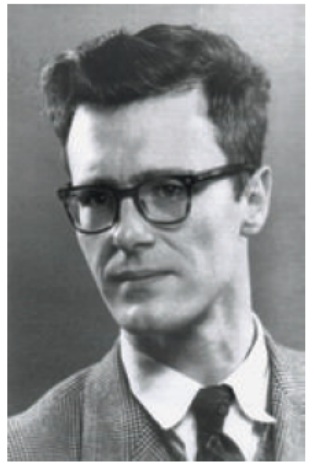
- Early programming languages
  - *Type checking*
  - *Recursion*
  - *Dynamic data structures*



Flowcharts for the "for" loop

Elements of modern programming languages

编程语言的发展



Edsger Dijkstra (1930–2002) was one of the pioneers of computer science. From the early days he was championing a mathematically rigorous approach to programming. In 1972 he received the Turing Award for his fundamental contributions to the development of programming languages. He was well known for his forthright opinions on programming and programming languages.

On FORTRAN:

> FORTRAN, "the infantile disorder", by now nearly 20 years old, is hopelessly inadequate for whatever computer application you have in mind today: it is now too clumsy, too risky, and too expensive to use.

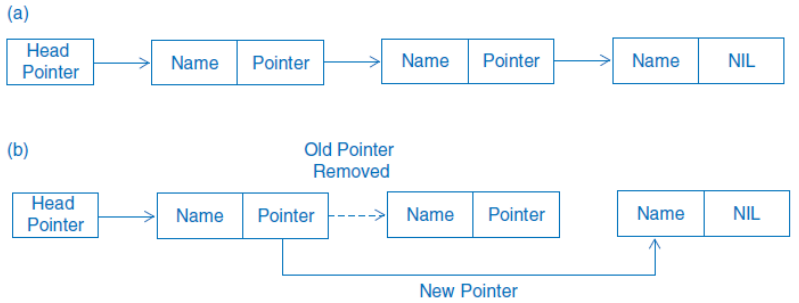
On COBOL:

> The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offense.

On BASIC:

> It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration.

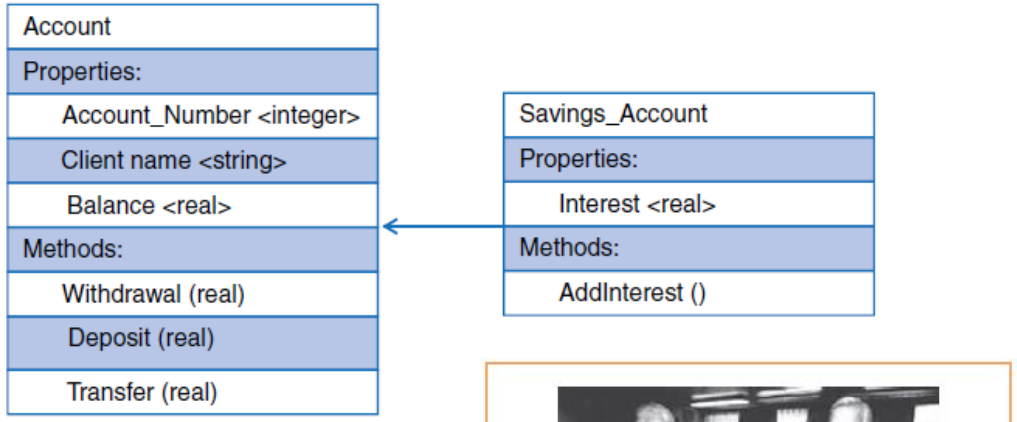
```
factorial (n)
    if n <= 1:
        return 1
    else:
        return n * factorial (n - 1)
```



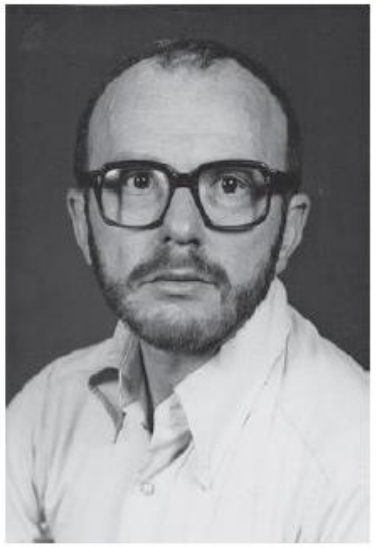
- *Recursion, the ability* for a program or subroutine to *call itself*
- *Dynamic data structures, for which* the memory space required is not set in advance but can change size as the program runs.

Recursion and dynamic data structures

编程语言的发展



Ole-Johan Dahl (1931–2002) (left) and Kristen Nygaard (1926–2002) were first to introduce classes and objects in their Simula programming language.

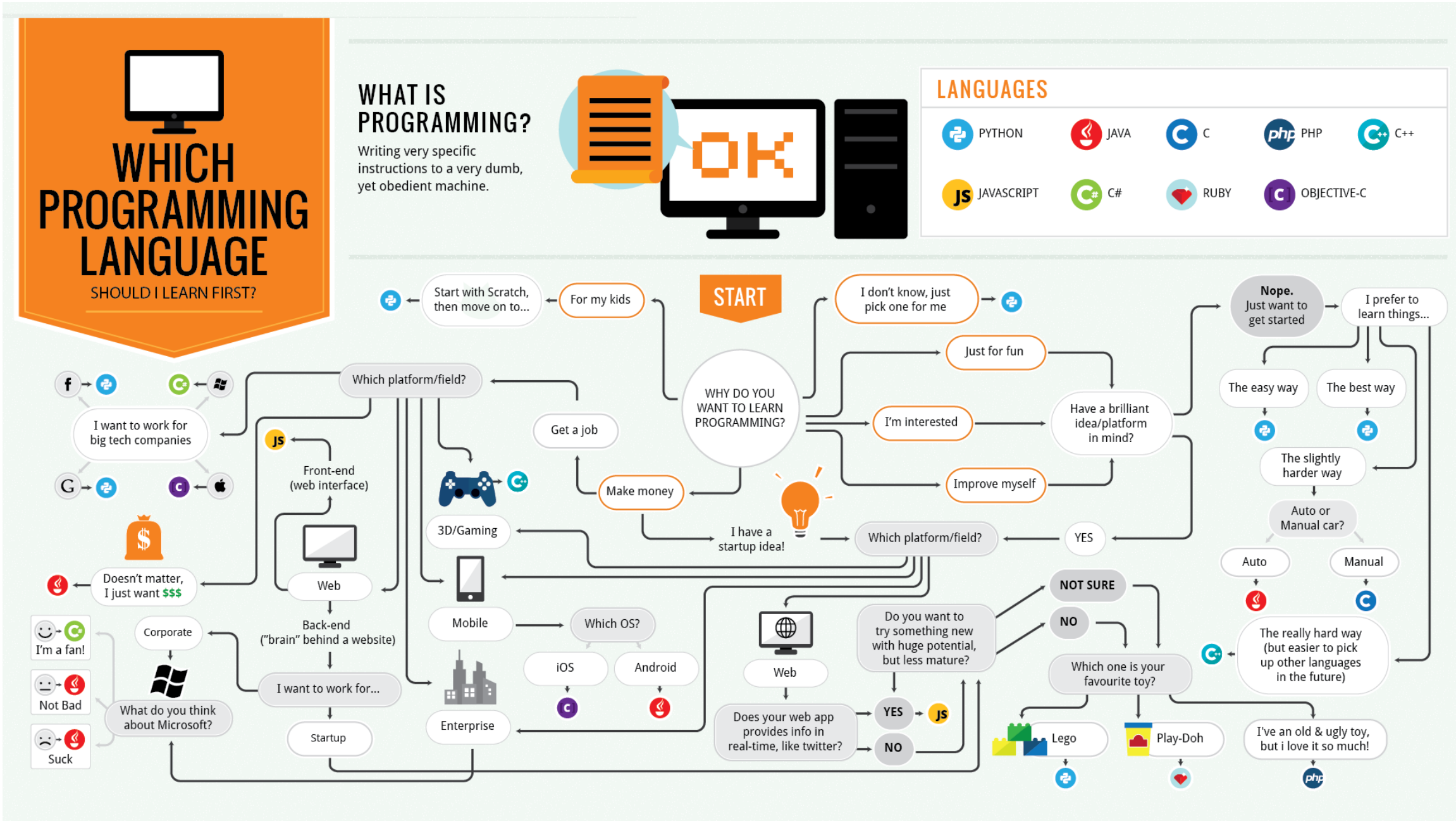


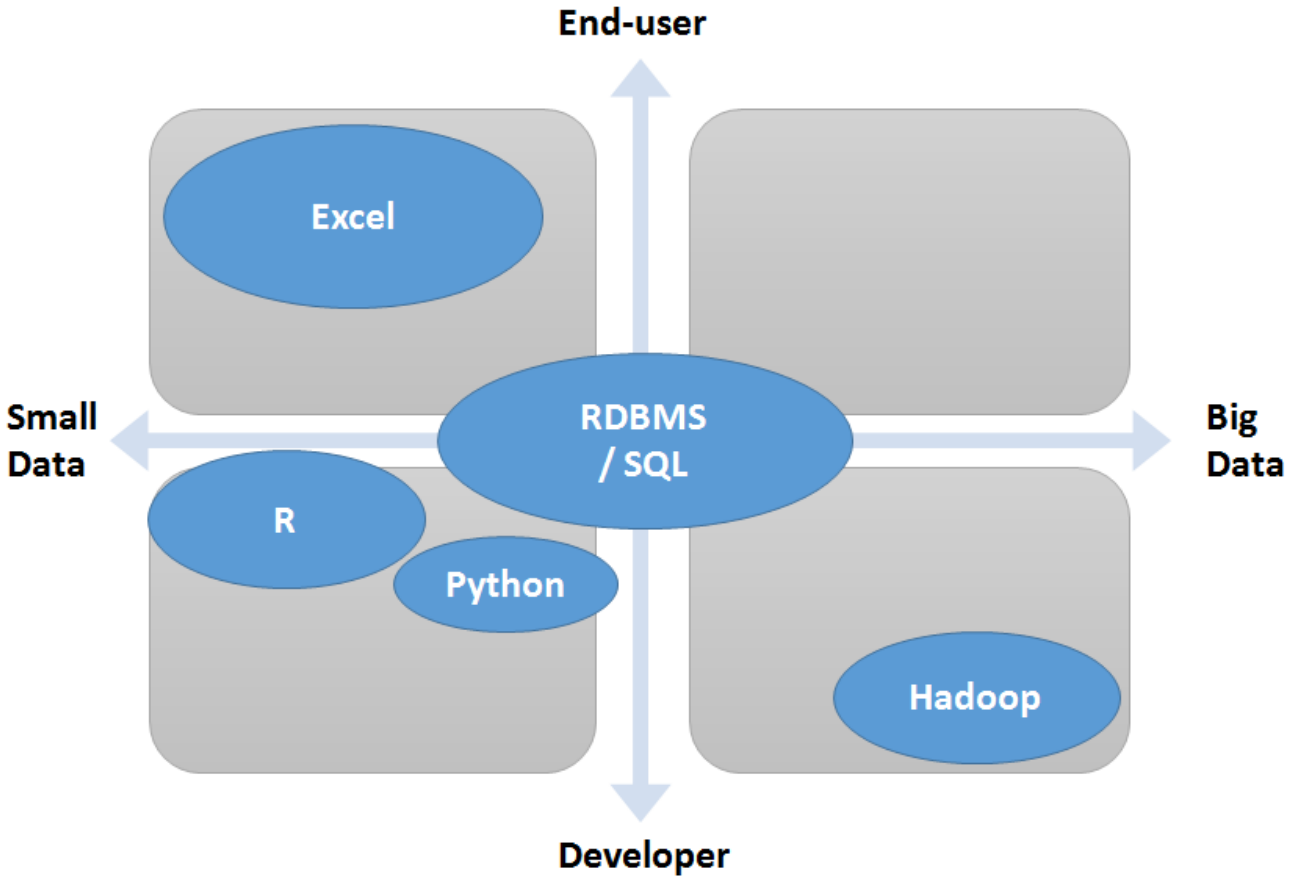
David Parnas is a Canadian computer scientist who pioneered ideas of "information hiding." These ideas are now an integral part of data abstraction in object-oriented programming.

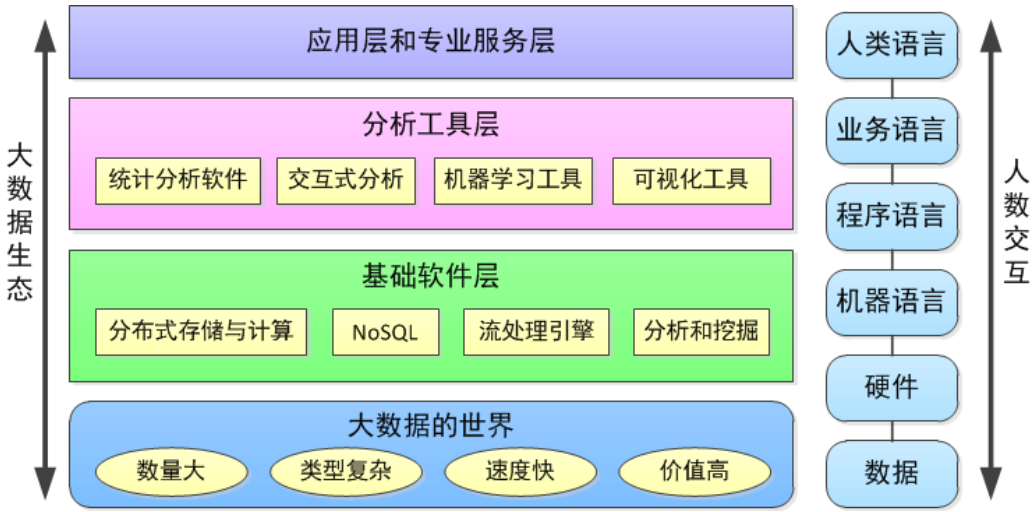
Bjarne Stroustrup designed and implemented the C++ programming language. Over the last two decades, C++ has become the most widely used language supporting object-oriented programming and has made abstraction techniques affordable and manageable for mainstream projects.

Programming with objects

编程语言的发展


WHICH PROGRAMMING LANGUAGE
SHOULD I LEARN FIRST?
WHAT IS PROGRAMMING?
Writing very specific instructions to a very dumb, yet obedient machine.
OK
LANGUAGES
PYTHON
JAVA
C
PHP
C++
JAVASCRIPT
C#
RUBY
OBJECTIVE-C
START
WHY DO YOU WANT TO LEARN PROGRAMMING?
For my kids
Start with Scratch, then move on to...
I don't know, just pick one for me
Just for fun
I'm interested
Improve myself
Have a brilliant idea/platform in mind?
Nope. Just want to get started
I prefer to learn things...
The easy way
The best way
The slightly harder way
Auto or Manual car?
Auto
Manual
YES
Which platform/field?
I have a startup idea!
Make money
Get a job
Which platform/field?
I want to work for big tech companies
Front-end (web interface)
Web
Back-end ("brain" behind a website)
Doesn't matter, I just want $$$
3D/Gaming
Mobile
Which OS?
iOS
Android
Enterprise
Corporate
I want to work for...
Startup
What do you think about Microsoft?
I'm a fan!
Not Bad
Suck
Web
Do you want to try something new with huge potential, but less mature?
NOT SURE
NO
Which one is your favourite toy?
Lego
Play-Doh
I've an old & ugly toy, but i love it so much!
The really hard way (but easier to pick up other languages in the future)
Does your web app provides info in real-time, like twitter?
YES
NO





面向数据科学的编程语言

SQL的强大功能

- 适用于数据分析的自然语言
  - 基于关系代数、面向集合
- 高效的语言
  - 声明式语言：直接表述想要的结果，而非获得方式
- 优化的处理
  - 将结果与方式脱钩有助于持续优化 SQL 引擎
- 持续创新
  - 内部处理、语言结构和数据访问一直在增强

面向数据科学的编程语言

SQL至关重要

“……对于我们业务逻辑的任一部分来说，处理非 ACID 数据存储都非常复杂，如果不用 SQL 查询，我们的业务根本没法开展。”

Google，VLDB 2013

Google

“[Facebook] 开始是用 Hadoop。现在我们在引入关系型数据库系统来增强 Hadoop…… [我们] 意识到使用错误的技术来解决某些问题可能比较困难。”

Ken Rudin，Facebook，TDWI 2013

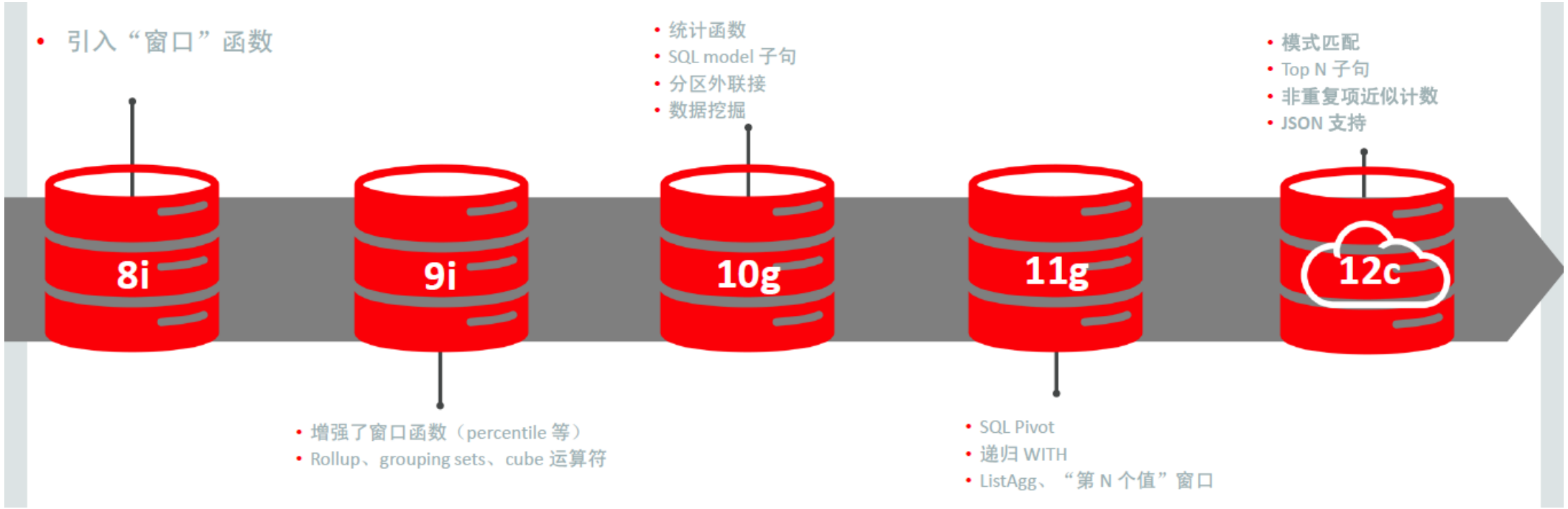
facebook

面向数据科学的编程语言

SQL广受欢迎

面向数据科学的编程语言

Oracle SQL：30 年创新之旅

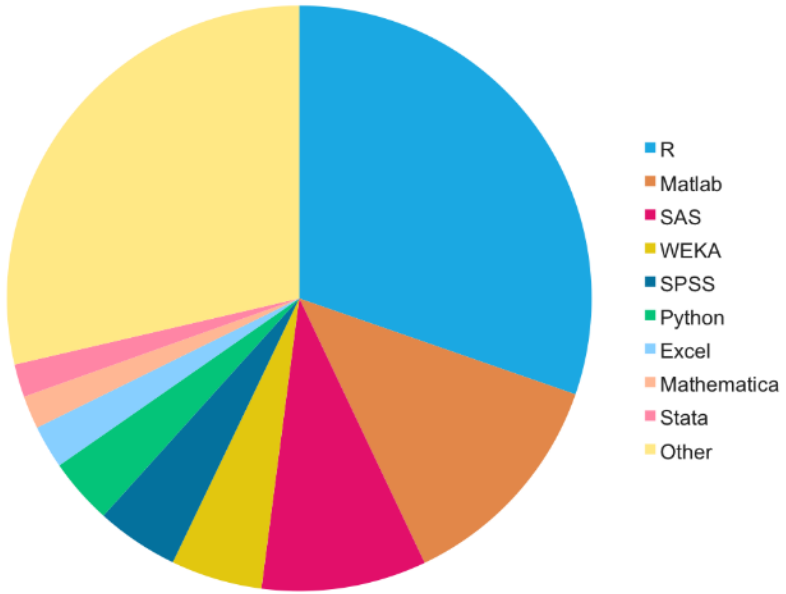


SQL语言不断改变

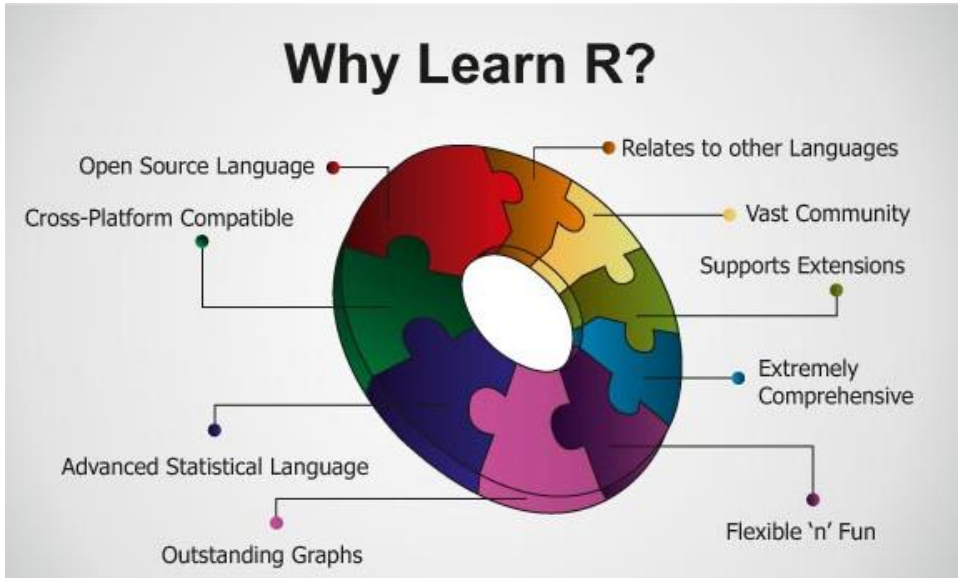
面向数据科学的编程语言

R 语言

R语言，一种自由软件编程语言与操作环境，主要用于统计分析、绘图、数据挖掘，内置多种统计学及数字分析功能。

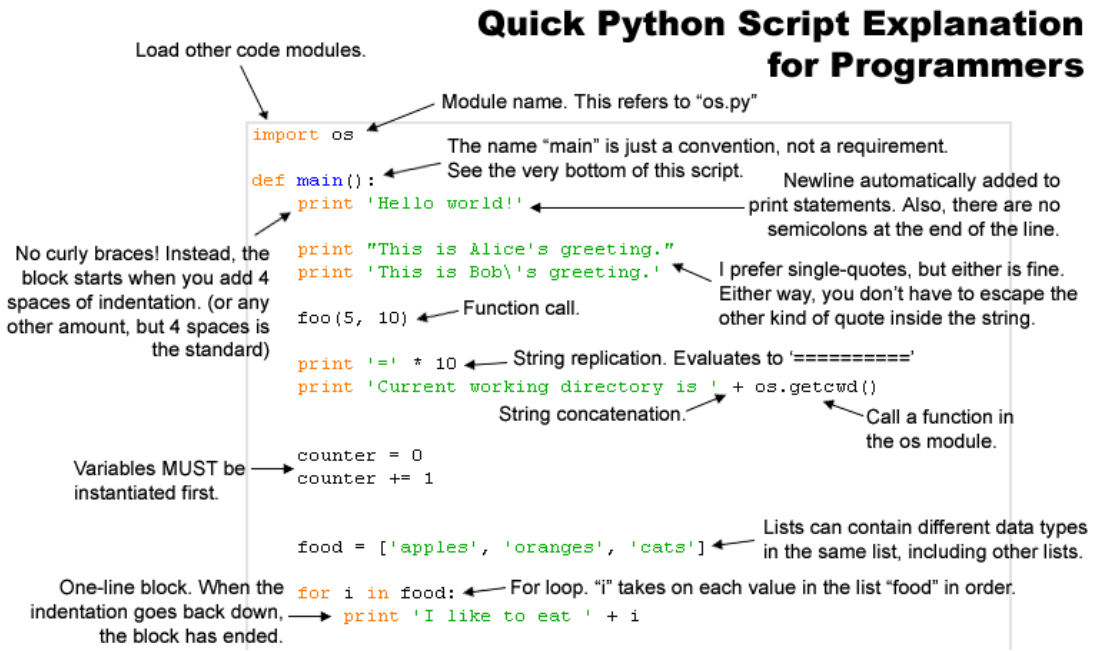


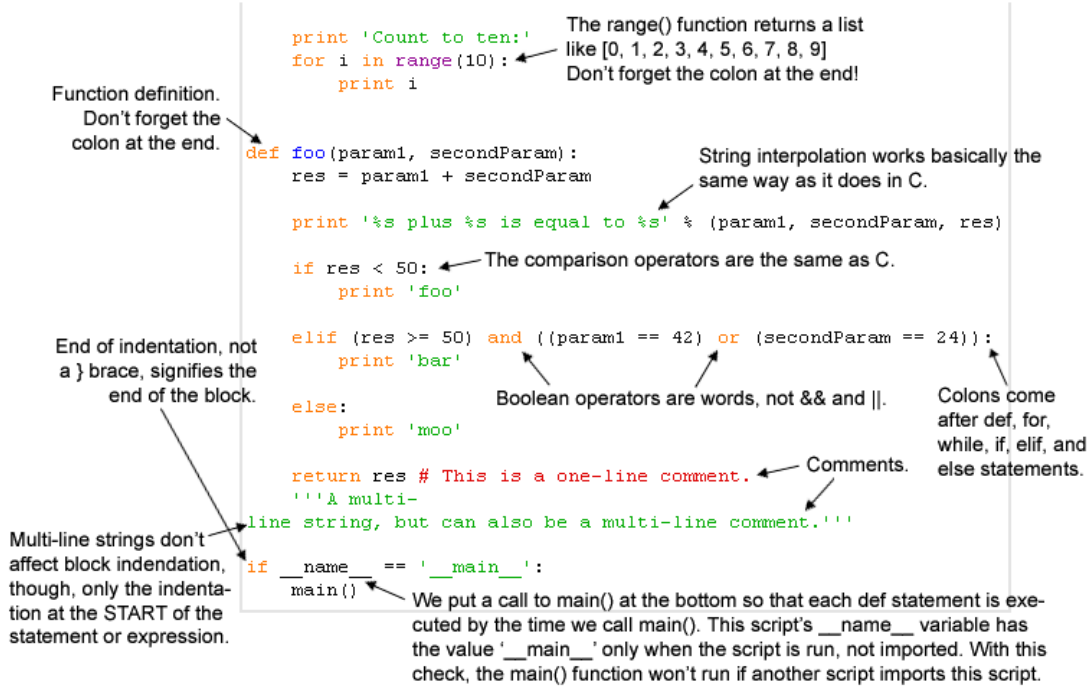
Popular programming languages on Kaggle



面向数据科学的编程语言

Python





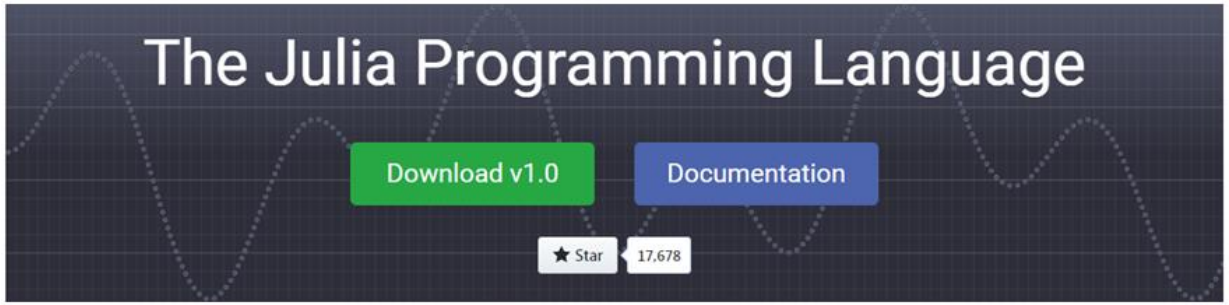
面向数据科学的编程语言

三种语言的比较

名称	特点	适用场景	招聘技能要求出现频数
R 语言	兼容性强，语言程序化也强，在编程语言方面需要投入的精力比 Python 大，但适用面较广	最常用数据分析工具之一，兼容性强	高频工具之一
Python	Life is too short，I use Python 以语言简单，注重数据分析的高效著称，尤其是在文本处理等数据结构化方面有很好优势	编程类数据分析，如文本字符等非结构化数据的处理	高频工具之一
SQL	数据库处理和分析的必备技能，属于数据库方面的基本工具	侧重数据库方面，如数据仓库等，作为 Oracle 等数据库方面的基础知识不可或缺	高频工具之一

面向数据科学的编程语言

数据科学的新星语言：Julia



Julia is fast!

Julia was designed from the beginning for high performance. Julia programs compile to efficient native code for multiple platforms via LLVM.

Dynamic

Julia is dynamically-typed, feels like a scripting language, and has good support for interactive use.

Optionally Typed

Julia has a rich language of descriptive datatypes, and type declarations can be used to clarify and solidify programs.

General

Julia uses multiple dispatch as a paradigm, making it easy to express many object-oriented and functional programming patterns. The standard library provides asynchronous I/O, process control, logging, profiling, a package manager, and more.

Technical

Julia excels at numerical computing. Its syntax is great for math, many numeric datatypes are supported, and parallelism is available out of the box. Julia's multiple dispatch is a natural fit for defining number and array-like datatypes.

Composable

Julia packages naturally work well together. Matrices of unit quantities, or data table columns of currencies and colors, just work — and with good performance.

面向数据科学的编程语言

第4章

数据的计算与程序表达

1 数据的计算

2 算法分析

3 数据结构与算法的关系

4 算法实例

5 计算机编程语言