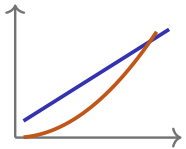
Algorithms & Programming

AP Computer Science Principles ▪ Unit 3

AP Computer Science Principles



What we will cover

1. Variables & data
2. Logic & conditionals
3. Iteration & algorithms
4. Lists & searching
5. Procedures & libraries
6. Randomness & limits

`total`

← 16

Variables, expressions & strings

Variable 变量

a named box;
assignment 赋值 ←

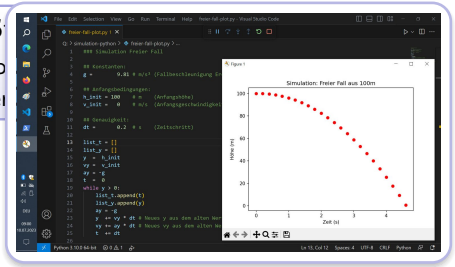
Expressions

× ÷ before + −:
$2+3\times4 = 14$

S

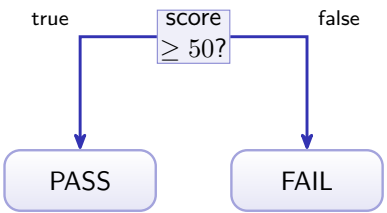
o
ter

Data abstraction 数据抽象 groups many values under one name – a **list 列表** – so the same code works for any amount of data.



Programs use variables, expressions, and control flow

Boolean logic & conditionals



A **Boolean expression** 布尔表达式 is true or false.

- **relational** 关系运算符: $>, <, =, \neq$
- **logical** 逻辑运算符: AND, OR, NOT

A **conditional** 条件语句 runs one branch – **selection** 选择. An **else-if** 否则如果 chain picks among three or more.

Boolean logic & conditionals

arithmetic

`+` `-` `*` `/`

`MOD` `DIV`

give a
number

relational

`=` `<>` `<` `>`

`<=` `>=`

compare →
true or **false**

logical

`AND` `OR`

`NOT`

join conditions →
true or **false**

Iteration & the three building blocks

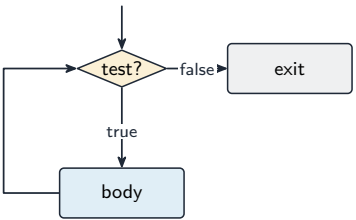
Iteration 迭代 repeats a block with a **loop** 循环 – beware the **infinite loop** 无限循环. Every **algorithm** 算法 is built from three structures:

Sequencing 顺序 | **Selection** | **Iteration**

Express an algorithm as **pseudocode** 伪代码, a **flowchart** 流程图, or natural language. Different algorithms can give the **same result** 相同结果.

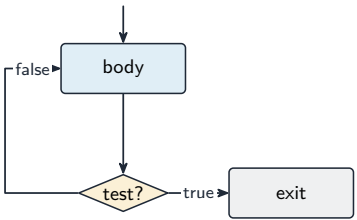
Iteration & the three building blocks

WHILE
(pre-condition)



tested first ⇒ may
run **zero** times

REPEAT
(post-condition)



tested last ⇒ runs
at least once

Lists & binary search

A **list** 列表 stores an ordered collection reached by **index** 索引 (from 0); **traverse** 遍历 it with a loop.

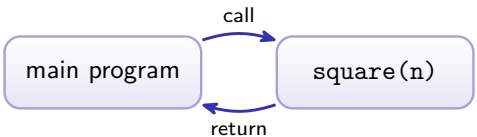
Binary search halves the range

Binary search 二分查找 needs a **sorted** 已排序 list: check the **middle** 中间, then keep only the half that could hold the target. Each step **halves** 减半 the range – 1000 items in ~ 10 checks, not 1000.

Search algorithms trade work for speed

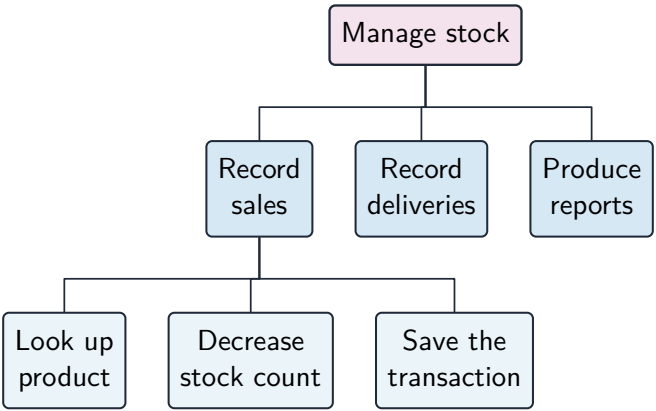
Lists & binary search

Procedures & libraries

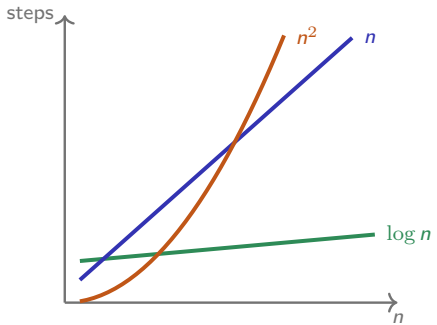


A **procedure** 过程 is reusable code you **call** 调用 by name, passing **arguments** 实参 into **parameters** 形参 and getting a **return value** 返回值. A **library** 库 bundles tested procedures – use its **API** 应用程序接口 without seeing inside. This is **procedural abstraction** 过程抽象.

Procedures & libraries



Algorithmic efficiency

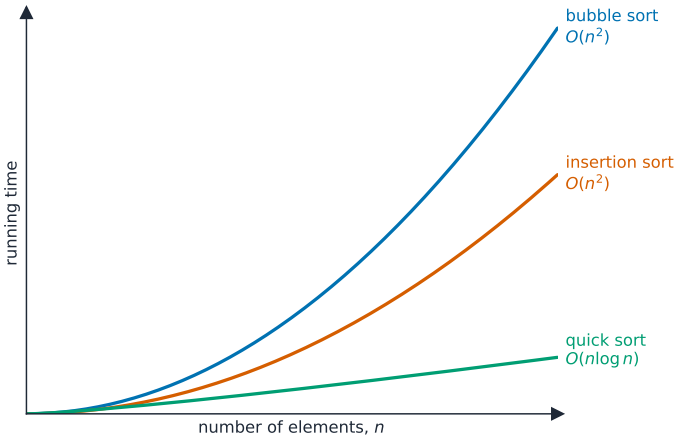


Algorithmic efficiency 算法效率 counts **steps** 步骤 as n grows.

- $\log n$, n, n^2 are **reasonable** 合理
- 2^n is **unreasonable** 不合理

When exact is too slow, use a **heuristic** 启发式方法.

Algorithmic efficiency



Randomness, simulation & undecidability

Random 随机 values add **variability** 可变性; a **simulation** 模拟程序 models a real **phenomenon** 现象 cheaply, but has **limitations** 局限性.

Decidable 可判定
an algorithm always answers (is n prime?)

Undecidable 不可判定
no algorithm for every case (will it halt?)

Undecidable means **impossible**, not merely **inefficient** 低效 (slow).

Key points

Algorithms = **sequencing, selection, iteration**

Binary search halves a sorted list each step

Procedures & **libraries** give abstraction & reuse

Efficiency compares growth; some problems are **undecidable**

Worked example – why halving wins

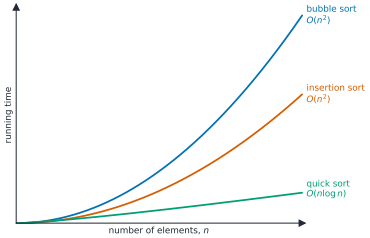
Search a sorted list of 8 items

Binary search halves the range each step: $8 \to 4 \to 2 \to 1$ – at most 3 comparisons ($\log_2 8 = 3$). A linear search could take up to 8. The advantage grows explosively: $1{,}000$ items need ≈ 10 binary steps (but up to $1{,}000$ linear); $1{,}000{,}000$ items need just ≈ 20.

Halving is what makes it a reasonable-time algorithm – and why the list must be sorted first.

Check yourself

1. Binary search on $1{,}000$ items: about how many steps?
2. What must be true of the list before binary search works?
3. Name the three building blocks of any algorithm.
4. What makes a problem undecidable?



Answers **1.** about 10 $(\log_2 1000)$ **2.** it must be sorted **3.** sequencing, selection, iteration **4.** no algorithm can solve it for every input