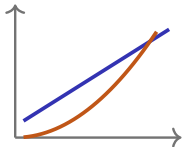


Algorithms & Programming

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

1

Variables & data

Variables, expressions & strings

Variable 变量

a named box;
assignment 赋值 $\leftarrow$

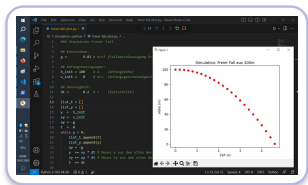
Expressions

$\times \div$ before $+-$:
 $2+3\times 4 = 14$

Strings 字符串

ordered characters;
index from 0

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

A variable is a named store whose value can change

age



a variable is a named box
that holds a value

Assignment replaces the stored value: after $\text{age} \leftarrow \text{age} + 1$, the box holds 17 and the 16 is gone.

The three families of operators

arithmetic

$+$ $-$ $*$ $/$

MOD

give a
number

relational

$=$ $\neq$ $<$

$>$ $\leq$ $\geq$

compare $\rightarrow$
true or false

logical

AND

OR

NOT

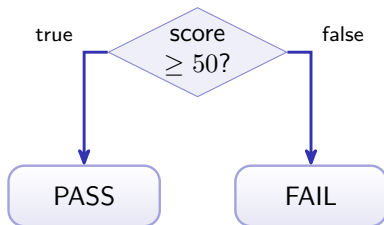
join conditions $\rightarrow$
true or false

The three families of operators: arithmetic, relational, and logical

2

Logic

Boolean logic & conditionals



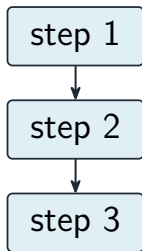
A **Boolean expression** 布尔表达式 is true or false.

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

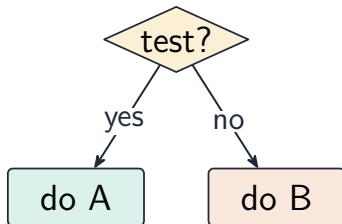
A **conditional** 条件语句 runs one branch – **selection** 选择. An **else-if** 否则如果 chain picks among three or more.

Selection chooses between paths based on a condition

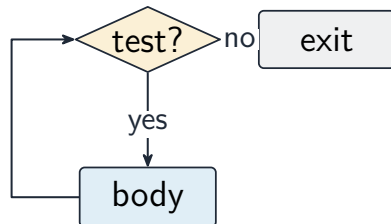
Sequence



Selection



Iteration



IF runs its block when the condition is true; ELSE gives the other path. Exactly one of the two runs.

3

Iteration & algorithms

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

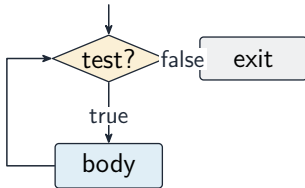
Selection

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.

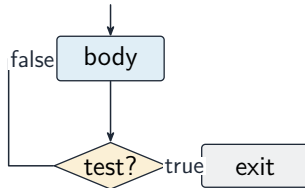
A pre-condition (WHILE) loop tests before the body

WHILE (pre-condition)



tested first $\Rightarrow$ may
run **zero** times

REPEAT (post-condition)



tested last $\Rightarrow$ runs
at least once

A pre-condition (WHILE) loop tests before the body, so it may run zero times

4

Lists & searching

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

A list holds many values in one variable

index	1	2	3	4	5
scores	90	75	60	88	50

↑
scores[2] is 75

A list holds many values in one variable, each found by its index

Binary Search

Diagram illustrating the binary search algorithm for finding the letter 'W' in an alphabet (A-Z).

Step 1: Initial range is A-Z (0 to 25). $low = 0$, $mid = 12$ (M), $high = 25$. Since $M < W$, go right.

Step 2: Range is M-Z (13 to 25). $low = 13$, $mid = 19$ (T), $high = 25$. Since $T < W$, go right.

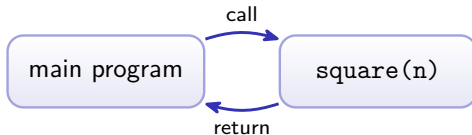
Step 3: Range is T-Z (20 to 25). $low = 20$, $mid = 22$ (W), $high = 25$. Since $W = W$, **found**.

Binary search halves the range at each step (the list must be sorted)

5

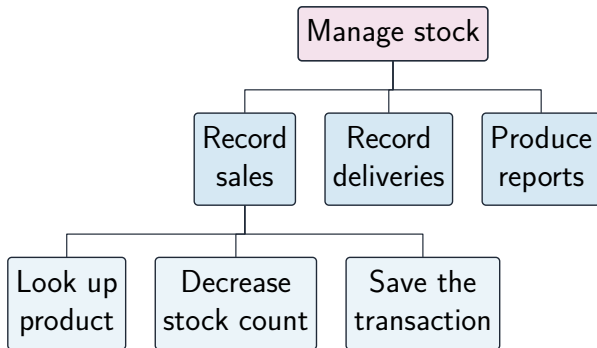
Procedures

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.

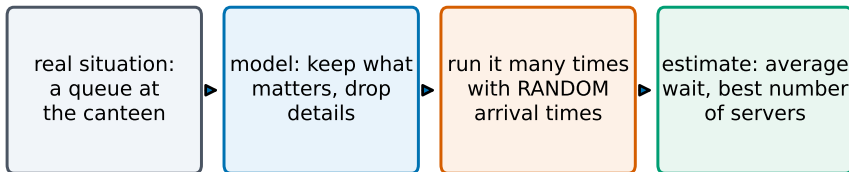
Developing Procedures



Decomposing a program into procedures and sub-procedures

Simulations

A **simulation** 模拟 is a program that models a real-world process to study it **safely** and **cheaply**.



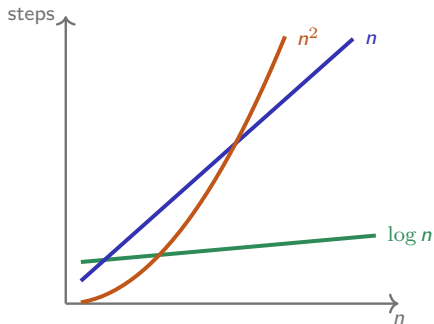
cheaper and safer than real experiments, but only as good as its assumptions

Simulations **simplify** reality (they leave out detail) and often use **randomness** to imitate chance events.

6

Randomness & limits

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.

The vocabulary this unit is built from

operators	+, -, *, /, and MOD – the remainder 余数 of a division: 17 MOD 5 is 2
concatenation 连接	joining strings end to end
conditional 条件 (selection)	choosing a path; a nested conditional 嵌套条件 puts one inside another
iteration 迭代	a loop – repeating a block
a procedure 过程 (function)	a named block, reusable and testable on its own
RANDOM(a, b)	returns a random integer from a to b inclusive , letting a program produce unpredictable results

Efficiency 效率 is how much time or memory an algorithm needs **as its input grows**. A **reasonable-time** algorithm's work grows like a polynomial of the input size.

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 ev-
ery case (will it halt?)

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

Undecidable problems

undecidable 不可判定

no algorithm can solve **every** case of the problem with a correct yes/no answer

the point

this is a **fundamental limit of computation**, not a limit of today's computers

the classic case

deciding whether an arbitrary program will halt

what is still possible

an algorithm may solve *some* instances, or give an answer that is usually right

the neighbouring idea

an **unreasonable-time** algorithm is solvable but grows too fast to run at scale

the distinction

undecidable means **no algorithm exists**; unreasonable means one exists and is too slow

Undecidable and intractable are different claims. The exam offers them as distractors for each other.

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**

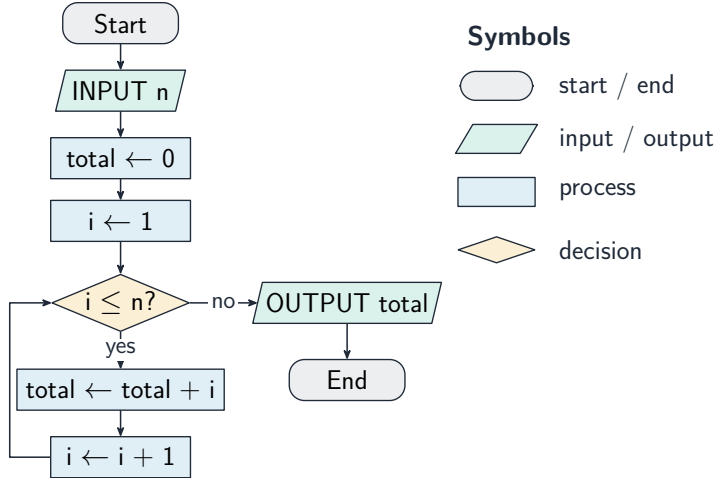
Worked example – why halving wins

Search a sorted list of 8 items

Binary search **halves** the range each step:
 $8 \rightarrow 4 \rightarrow 2 \rightarrow 1$ – 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 .

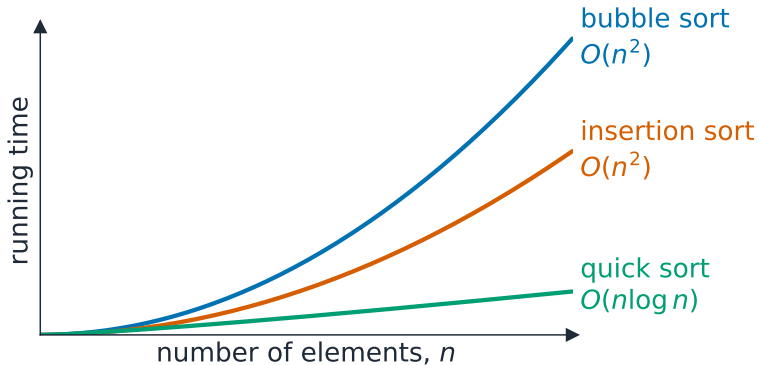
Halving is what makes it a **reasonable-time** algorithm – and why the list must be **sorted** first.

Developing Algorithms



A flowchart lays out an algorithm using the standard symbols

Check yourself, in detail



for large n : $O(n \log n)$ (quick/merge) beats $O(n^2)$
(bubble/insertion)

How the running time of an algorithm grows with the input size n

Exam tips

- Know a variable is a named store for a value and trace how **assignment** updates it step by step.
- Read the AP **pseudocode** carefully – a `<-` expression assigns, and lists are **1-indexed** on the exam reference sheet.
- Distinguish a variable from a **list** (a collection accessed by index) and use list operations correctly.
- Evaluate expressions with the right precedence and boolean logic (AND, OR, NOT).
- Pick clear, meaningful variable names – the written tasks reward readable code.

7

Recap

Unit 3 – key takeaways

1

Variable

named store; trace assignment
step by step

2

Pseudocode

$a \leftarrow \text{expression}$; lists are 1-
indexed

3

List

a collection by index – not a
single variable

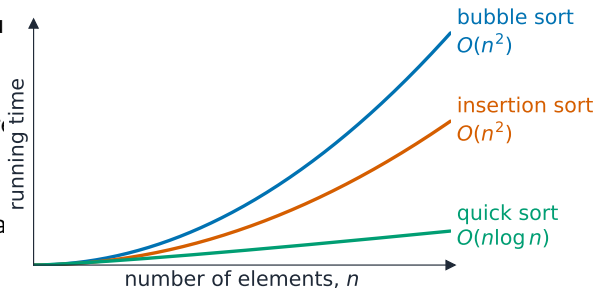
4

Names

clear names; written tasks reward
readable code

Check yourself

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



for large n : $O(n \log n)$ (quick/merge) beats $O(n^2)$ (bubble/insertion)

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