

Exercise walk-through

4.4

Algorithms

GAC Mathematics

1

The method

Method — Key knowledge: a correct trace follows the stated step order and loop bounds

An **algorithm** 算法 states unambiguous steps for a task over defined allowed inputs. To solve the stated task it must produce the required result and terminate under those input conditions. A **flowchart** 流程图 uses labelled flow arrows, process boxes and decision branches; input/output uses a parallelogram.

Pseudocode 伪代码 describes the procedure without promising a particular language's exact syntax. On this sheet, assignment updates the left-hand variable, FOR bounds are inclusive, and MOD is the nonnegative remainder for nonnegative integers and a positive divisor. Trace variables after each stated update. Binary search requires an ordered list and consistent index/bound conventions; linear search can stop at an early match and does not always inspect every item.

Method — Worked example: repeat a remainder step until the second number is zero

Known: start with positive integers a equal to 10 and b equal to 6. While b is nonzero, compute r as $a \bmod b$, then set a to b and b to r . Pairs after complete iterations are $(6,4)$, $(4,2)$, $(2,0)$, giving output 2. Temporary r preserves the remainder before a and b change. Each nonzero remainder is smaller than the previous positive b , supporting termination.

Exercises 2.1·2.2

2.1 Start total at 0. For i equal to 1, 2 and 3, update total by adding i . List the total after each update and the final output. [2]

2.2 A linear minimum-finding procedure starts smallest at the first value of $[5, 1, 4]$, then compares each remaining value with smallest. List the two later smallest values and the final minimum. [2]

Exercises 2.1-2.2

2.1 Start total at 0. For i equal to 1, 2 and 3, update total by adding i . List the total after each update and the final output. [2]

2.1

Successive totals are 1, 3 and 6. Correct repeated-update method earns M1. Final output 6.

2.2 A linear minimum-finding procedure starts smallest at the first value of $[5, 1, 4]$, then compares each remaining value with smallest. List the two later smallest values and the final minimum. [2]

2.2

After comparing 1, smallest becomes 1; comparing 4 leaves it at 1. Correct update/retain sequence. Final minimum 1.

Exercises 2.3·2.4

2.3 Draw a flowchart that inputs one real number n , outputs “nonnegative” if n is at least zero and “negative” otherwise, then stops. Label the decision branches. [3]

2.4 Binary search uses zero-based indices, inclusive low/high bounds and $\text{floor}((\text{low}+\text{high})/2)$. Trace its middle indices while finding 14 in the sorted list $[2, 5, 8, 11, 14, 17, 20]$, starting $\text{low} = 0$ and $\text{high} = 6$. [3]

3 Exam-style questions

Exercises 2.3-2.4

2.3 Draw a flowchart that inputs one real number n , outputs “nonnegative” if n is at least zero and “negative” otherwise, then stops. Label the decision branches. [3]

2.3

Input n followed by a decision testing $n \geq 0$. The yes branch outputs nonnegative and the no branch negative; both are labelled correctly. Both output branches reach a stop, with coherent arrows and appropriate process/decision/input-output symbols. Zero follows the nonnegative branch.

2.4 Binary search uses zero-based indices, inclusive low/high bounds and $\text{floor}((\text{low} + \text{high})/2)$. Trace its middle indices while finding 14 in the sorted list $[2, 5, 8, 11, 14, 17, 20]$, starting $\text{low} = 0$ and $\text{high} = 6$. [3]

3 Exam-style questions

2.4

First middle is 3, value 11. Since 14 is greater, set low to 4; next middle is 5, value 17. Set high to 4; next middle is 4, value 14, found. The compared middle indices are 3, 5, 4.

Exercise 3.1

3.1 A procedure is intended to add integers 1 through n for a nonnegative integer n . It starts $\text{total} = 0$ and $i = 1$, repeats while $i < n$, adds i to total , then increments i . (a) Trace it for $n = 4$. (b) Identify the boundary error and correct the condition. (c) State the correct outputs for $n = 4$ and $n = 0$. [6]

Exercise 3.1

3.1 A procedure is intended to add integers 1 through n for a nonnegative integer n . It starts $\text{total} = 0$ and $i = 1$, repeats while $i < n$, adds i to total , then increments i . (a) Trace it for $n = 4$. (b) Identify the boundary error and correct the condition. (c) State the correct outputs for $n = 4$ and $n = 0$. [6]

3.1

- (a) The original loop adds 1, 2 and 3. It outputs 6 rather than 10.
- (b) The strict condition excludes n . Use i less than or equal to n .
- (c) Correct output for 4 is 10. For 0, the corrected loop executes zero times and outputs the initial total 0. The input restriction excludes negative and noninteger n .

Exercise 3.2

3.2 Use the worked Euclidean procedure for $a = 48$ and $b = 18$. (a) Trace a , b and r through all complete iterations and give the output. (b) Explain why r must be computed before overwriting the old inputs. (c) Justify termination for positive integer inputs and explain what the output represents. (d) State how the procedure handles a positive a with b initially zero if that case is explicitly allowed. [8]

Exercise 3.2

3.2 Use the worked Euclidean procedure for $a = 48$ and $b = 18$. (a) Trace a , b and r through all complete iterations and give the output. (b) Explain why r must be computed before overwriting the old inputs. (c) Justify termination for positive integer inputs and explain what the output represents. (d) State how the procedure handles a positive a with b initially zero if that case is explicitly allowed. [8]

3.2

(a) First remainder is $48 \text{ MOD } 18 = 12$, giving pair $(18,12)$. Next remainder $18 \text{ MOD } 12 = 6$ gives $(12,6)$. Final remainder $12 \text{ MOD } 6 = 0$ gives $(6,0)$, so output 6.

(b) The remainder must use the old a and b ; overwriting them first changes the division. A temporary r preserves it.

(c) For positive b , the new nonnegative remainder is strictly smaller than b . A strictly descending sequence of nonnegative integers reaches zero, so the procedure terminates. Euclidean remainder replacement preserves common divisors, and the final output is the greatest common divisor.

(d) The zero test immediately outputs a without taking MOD by zero.