Exercise walk-through

4.4 Algorithms

GAC Mathematics

Sets, relations and functions (2)

- A **relation** pairs elements of two sets. A **function** is a **relation** in which each input has exactly one output. "Each student $\to$ their birthday" is a **function** (two students may share a birthday); "each birthday $\to$ the students born on it" is not, because one input can have several outputs.
- A **Venn diagram** , introduced by the English logician John Venn in 1880, turns a set problem into a picture. Check that the regions add up to the total.

$$N_{neither} = |U| - |F \cup G| = 30 - 26 = 4$$

Counting systems

- A **number base** b uses the digits 0 to $b-1$, and each place is worth a power of b. **Decimal** uses ten, **binary** two, **hexadecimal** sixteen (digits 0–9 then A–F).
- Every digit's value is its **place value**: in binary the places are 1, 2, 4, 8, 16 and so on.
- **Hexadecimal** is shorthand for binary: one hex digit is exactly four bits. Web colours use it: ‘#FF8000‘ means red 255, green 128, blue 0, a bright orange.

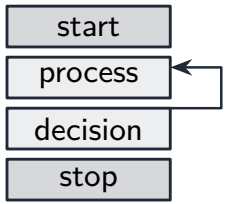
$$V = \sum d_i 2^i$$

Graphs and networks

- A **graph** is a set of **vertices** joined by **edges**. It models anything with connections: roads, friendships, dependencies.
- The **degree** of a vertex is the number of **edges** that meet it. Every edge has two ends, so the degrees always add up to twice the number of edges. This is the fact behind Euler's Königsberg proof.
- A **tree** is a connected **graph** with no cycles; a **tree** with n vertices has $n - 1$ edges. A family tree and a computer's folder structure are trees.

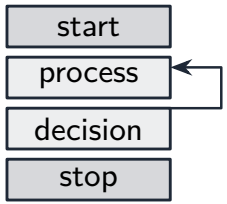
Exercise 2.1

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]



Exercise 2.1

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]

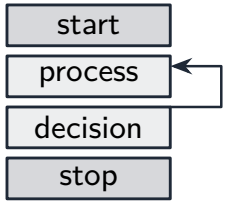


2.1

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

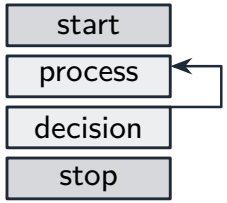
Exercise 2.2

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]



Exercise 2.2

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]



2.2

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

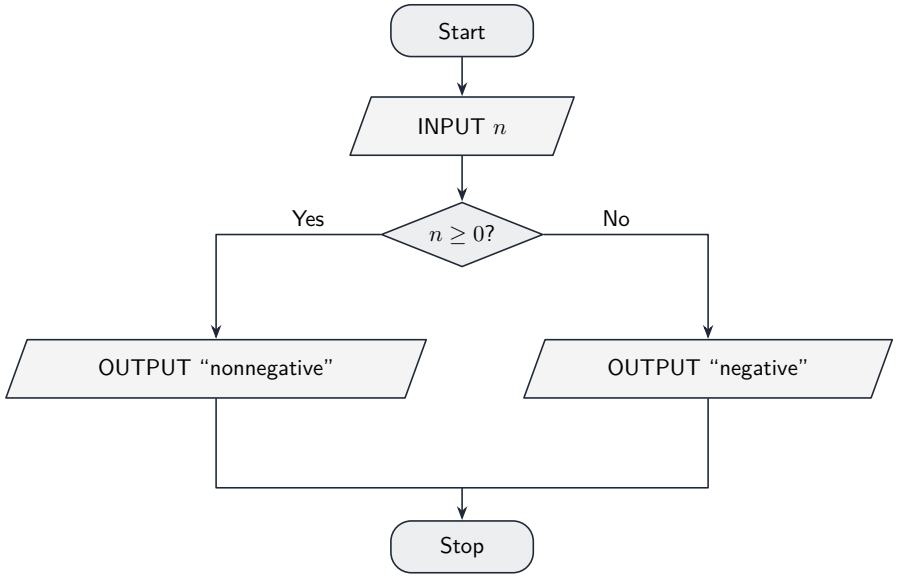
Exercise 2.3

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]

Exercise 2.3

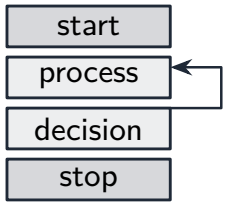
2.3 Draw a flowchart that inputs one real number n, outputs “nonnegative” if n is at least zero and “negative” otherwise, then stops. Label the decision branches. [3]

2.3 Flowchart



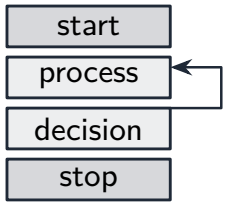
Exercise 2.4

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



Exercise 2.4

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

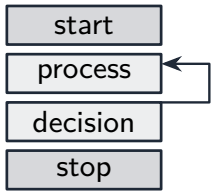


2.4

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.

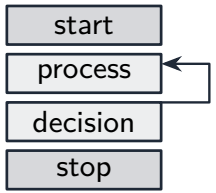
Exercise 3.1

3.1 A procedure is intended to add integers 1 through n for a nonnegative integer n. It starts 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 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

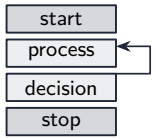
(a) The original loop adds 1, 2 and 3. It outputs 6 rather than 10.

(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.

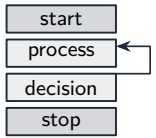
Exercise 3.2

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 MOD 18 = 12, giving pair (18,12). Next remainder 18 MOD 12 = 6 gives (12,6). Final remainder 12 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.

exercises.lol exercises.com.cn