tags: - cocubes - coding - solutions

# Coding Solutions and Methods

## 1. Unit digit of a power

Only the last digit of x matters. Last digits repeat in a short cycle.

Algorithm:

1. If y = 0, answer is 1.
2. Let d = x MOD 10.
3. Use the last-digit cycle for d.
4. Select position y MOD cycleLength; use final cycle entry when remainder is zero.

Example: 7 has cycle 7, 9, 3, 1. Unit digit of 7^103 is the third entry because 103 MOD 4 = 3.

## 2. Card total

Initialise total = 0 and positiveCount = 0. Scan values once. Add every value to total; if value > 0, increment positiveCount.

## 3. First unique character

1. Scan string and count each character.
2. Scan again in original order.
3. First character with count 1 is answer.

Two scans preserve original ordering.

## 4. Pair with target sum

Scan left to right. For each x, needed = T − x. If needed is already in a set, answer exists. Otherwise add x.

Why it works: every earlier value is stored, so when the later member of a valid pair is reached, its complement is found.

## 5. Valid palindrome

Set left at start and right at end.

1. Skip non-alphanumeric characters at both ends.
2. Compare lowercased characters.
3. If unequal, false.

4. Move both inward until they cross.

## 6. Balanced brackets

Map closing bracket to matching opening bracket.

1. Push each opening bracket.
2. On closing bracket, stack must be nonempty and top must match; otherwise false.
3. End result is true only when stack is empty.

## 7. Second largest distinct

Maintain largest and secondLargest.

For each x:

- if x > largest: secondLargest = largest; largest = x;
- else if x is distinct from largest and x > secondLargest: secondLargest = x.

Use a sentinel/optional values carefully to support negatives.

## 8. Equal-half digit substring

For every even length L from largest down to 2:

1. For every starting index, compare the digit sum of first L/2 characters with sum of next L/2.
2. Return the first valid L.

Use prefix sums to obtain any digit-range sum quickly. For a two-day assessment, write a simple correct version if constraints are modest; optimise only if constraints require it.

## 9. Count subarrays with sum K

Let prefix be sum so far. A previous prefix equal to prefix − K identifies a subarray summing to K.

1. Start map with count[0] = 1.
2. For each x: prefix += x.
3. Add count[prefix − K] to answer.
4. Increment count[prefix].

## 10. Grid shortest path

Use BFS:

1. Enqueue start with distance 0 and mark visited.
2. Repeatedly remove front cell.
3. Explore valid unblocked unvisited up/down/left/right neighbours.

4. Enqueue them with distance + 1.
5. First time target is removed/reached gives shortest path.

[[05_Coding/Coding_Playbook|Coding playbook]] · [[05_Coding/Pattern_Catalog|Pattern catalog]]