Quantitative Reasoning & Interview Preparation

Informal Notes for Problems on Probability, Estimation, and Financial Intuition

Ryan McMillan, MCompSci (cand.)

Master of Computer Science — University of New England

January 13, 2026

Contents

1 Introduction

This document is a curated portfolio of algorithmic solutions. Each selected problem includes:

- a brief problem summary,
- the key idea,
- an algorithm outline,
- a correctness sketch,
- time and space complexity,
- a reference C++ implementation.

The focus is on clarity, correctness, and reasoning.

2 Arrays & Hashing

2.1 Demo 0000: Triangular Number (Pipeline Test)

Problem. Given an integer $n \geq 0$, compute

$$T(n) = \sum_{i=1}^{n} i.$$

Return $T(n)$.

Key idea. Use the closed-form identity

$$T(n) = \frac{n(n+1)}{2}.$$

This avoids iteration and directly demonstrates asymptotic reasoning.

Algorithm. Compute $\frac{n(n+1)}{2}$ using integer arithmetic.

Correctness sketch. We use the classical identity:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Therefore the algorithm returns exactly $T(n)$ for all $n \geq 0$.

Complexity. Time: $O(1)$, Space: $O(1)$. For comparison, the iterative approach

$$\sum_{i=1}^{n} i$$

runs in $O(n)$ time and $O(1)$ space. As $n \to \infty$, the closed-form approach dominates asymptotically.

```cpp
#include <bits/stdc++.h>
using namespace std;

/*
  Demo problem for the portfolio PDF pipeline.

  Task: Given n >= 0, compute sum_{i=1..n} i = n(n+1)/2.

  Purpose: Exercise code inclusion and asymptotic discussion.
*/

long long triangular_number(long long n) {
    // O(1) time, O(1) space
    return n * (n + 1) / 2;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    long long n;
    if (!(cin >> n)) return 0;
    cout << triangular_number(n) << "\n";
    return 0;
}
```

Listing 1: Reference implementation (C++) for the triangular number problem.