

# **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

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Arrays &amp; Hashing</b>	<b>2</b>
2.1	Demo 0000: Triangular Number (Pipeline Test) . . . . .	2

## 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  $n \rightarrow \infty$ , the closed-form approach dominates asymptotically.

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5  Demo problem for the portfolio PDF pipeline.
6
7  Task: Given  $n \geq 0$ , compute  $\sum_{i=1..n} i = n(n+1)/2$ .
8
9  Purpose: Exercise code inclusion and asymptotic discussion.
10 */
11
12 long long triangular_number(long long n) {
13     //  $O(1)$  time,  $O(1)$  space
14     return n * (n + 1) / 2;
15 }
16
17 int main() {
18     ios::sync_with_stdio(false);
19     cin.tie(nullptr);
20
21     long long n;
22     if (!(cin >> n)) return 0;
23     cout << triangular_number(n) << "\n";
24     return 0;
25 }
```

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