Two Sum with Two-End Trim: Toward Logarithmic Squared Complexity

Zhikai Wang zhikai.wang@alumni.ucla.edu

May 2025

1 Introduction

The classic Two Sum problem asks whether there exist two numbers in an array that sum to a given target t. In this work, we focus on the variant where the input array is sorted, and the target is guaranteed to exist. We propose a refined approach called *Two-End Trim* and extend it with an *Opportunistic Guessing* strategy that improves worst-case behavior.

2 Two-End Trim

Problem Definition

Given a sorted array a of length $l \geq 2$, and a target t, find two indices $i < j$ such that $a[i] + a[j] = t$. It is guaranteed that such a pair exists and that:

- $a[0] + a[1] \leq t$, otherwise the smallest sum will overflow the target
- $a[l-2] + a[l-1] \geq t$, otherwise the largest sum will underflow the target

Algorithm: Two-End Trim

Start with $i = 0$ and $j = l - 1$.

1. **findRight**: Trim the right end.
   - Find the largest index $nj > i$ such that $a[nj] + a[i] \leq t$.
   - If $a[nj] + a[i] = t$, return $[i + 1, nj + 1]$.
   - Set $j = nj$.
2. **findLeft**: Trim the left end.
   - Find the smallest index $ni < j$ such that $a[ni] + a[j] \geq t$.
   - If $a[ni - 1] + a[j] = t$, return $[ni, j + 1]$.
   - Set $i = ni$.
3. Repeat until solution is found.

Edge cases:

- If $a[0] + a[1] > t$: return -1.
- If $a[l-2] + a[l-1] < t$: return ∞.

Analysis

Best case: $\mathcal{O}(1)$. Worst case: $\mathcal{O}(k \log n)$ where $k = \min(a_i, l - b_i)$, given $a + b = t$, a_i and b_i are their indices.

3 Two-End Trim Worst Case Study

We observe that the worst case occurs when a_i and b_i are both around the middle of the array. For instance, when $a_i \approx b_i \approx l/2$, the minimum distance k is maximized.

4 Two-End Trim with Opportunistic Guessing

To improve worst-case complexity toward $\mathcal{O}(\log^2 n)$, we propose an opportunistic guessing step:

- After each `findRight` and `findLeft`, attempt to guess the position of the midpoint near $t/2$.
- Compute the nearest index to the value $t/2$.
- From that guess ($newI$, $newJ$), move inward by a fixed percentage (e.g., 50%) to derive $newI_2$ and $newJ_2$. We can do multiple sampling guesses, eg 5% to 95%, even we can correctly guess 1% each loop, the total loop count will subject to $\mathcal{O}(\log n)$.
- Perform `findLeft` and `findRight` again on this reduced scope.
- If the guess is valid (i.e., leads to a correct sum), return. Otherwise, roll back and continue the trimming.

Observation

In practice, this opportunistic guessing improves worst-case performance significantly. Our experimental code confirms a drop from linear worst-case steps (e.g., 500 for $n = 1000$) to under 10, aligning with the $\log^2 n$ target. Here we just name it opportunistic, in fact, there is sound reasoning behind this. When the two-end-trim turns slow, each loop moves the index by 1 or 2 only, we know from the analysis in former section, a and b are pushed toward each other index-wise. Two-end-trim is rigid and fit for sparse indices distribution, the guess is based on former trim and fit for dense indices distribution.

5 Conclusion

We proposed a refined two-pointer approach called Two-End Trim and further enhanced it with an opportunistic guessing mechanism. Theoretical reasoning and sample data show a significant reduction in worst-case iterations, with potential to formalize this improvement further in complexity terms.

6 Further Research

An immediate application of this problem can be for Goldbach's conjecture as computer validation in limited scope. A large enough even number is the sum of two prime numbers. Given a large even number t and a complete list of prime numbers in scope, we can search for two prime numbers one is less than $t/2$ and the other is greater than $t/2$.