

CAPSTONE PROJECT

Count Good Triplets in an Array

**CSA0695- DESIGN ANALYSIS AND ALGORITHMS
FOR OPEN ADDRESSING TECHNIQUES**

SAVEETHA SCHOOL OF ENGINEERING

SIMATS ENGINEERING



Supervisor

Dr. R. Dhanalakshmi

Done by

K. Priyanka (192210196)

Count Good Triplets in an Array

PROBLEM STATEMENT:

Given two 0-indexed arrays `nums1` and `nums2` of length `n`, both of which are permutations of `[0, 1, ..., n - 1]`. A good triplet is a set of 3 distinct values which are present in increasing order by position both in `nums1` and `nums2`. In other words, if we consider `pos1v` as the index of the value `v` in `nums1` and `pos2v` as the index of the value `v` in `nums2`, then a good triplet will be a set (x, y, z) where $0 \leq x, y, z \leq n - 1$, such that $pos1x < pos1y < pos1z$ and $pos2x < pos2y < pos2z$. Return the total number of good triplets.

Example 1:

Input: `nums1 = [2,0,1,3]`, `nums2 = [0,1,2,3]`

Output: 1

Explanation:

There are 4 triplets (x,y,z) such that $pos1x < pos1y < pos1z$. They are $(2,0,1)$, $(2,0,3)$, $(2,1,3)$, and $(0,1,3)$. Out of those triplets, only the triplet $(0,1,3)$ satisfies $pos2x < pos2y < pos2z$. Hence, there is only 1 good triplet

ABSTRACT:

This project is to determine the number of good triplets in two given permutations of integers. A good triplet is defined as a set of three distinct values found in both arrays in increasing order by their indices.

INTRODUCTION:

Given two 0-indexed arrays `nums1` and `nums2`, each of length `n`, where both arrays are permutations of the set $[0, 1, \dots, n-1]$, we are interested in counting the number of "good triplets". A triplet (x, y, z) is defined as good if it satisfies the following conditions:

1. **Distinct Values:** The indices x, y , and z must be distinct.
2. **Increasing Order in `nums1`:** The values at these indices must appear in increasing order in `nums1`. Specifically, if v_1, v_2 , and v_3 are the values at indices x, y , and z , respectively, then $\text{pos1}[v_1] < \text{pos1}[v_2] < \text{pos1}[v_3]$ where $\text{pos1}[v]$ denotes the index of value v in `nums1`.
3. **Increasing Order in `nums2`:** Similarly, the values at these indices must also appear in increasing order in `nums2`, which means $\text{pos2}[v_1] < \text{pos2}[v_2] < \text{pos2}[v_3]$ where $\text{pos2}[v]$ denotes the index of value v in `nums2`.

The goal is to return the total number of such good triplets.

Understanding the Problem:

- **Permutations:** Both `nums1` and `nums2` are permutations of the same set, meaning each contains all integers from 0 to `n-1` exactly once. This ensures that each integer's position is unique in each array.
- **Good Triplet:** A triplet is considered good if, when ordered by their positions in both `nums1` and `nums2`, they maintain a strict increasing order. This dual-order constraint makes the problem non-trivial and interesting.

Objective:

The primary objective is to efficiently count all possible triplets (x, y, z) that satisfy the above conditions, leveraging the properties of permutations and orderings. This problem requires careful consideration of indexing and order properties, often necessitating sophisticated algorithms to handle potentially large input sizes effectively.

Approach:

To address this problem, one can use a combination of position mappings and data structures to count valid triplets efficiently. By mapping each value to its index in both permutations and analyzing the order constraints, we can systematically identify and count all good triplets. Advanced techniques like Fenwick Trees or Segment Trees may be employed to facilitate efficient counting and aggregation operations.

In summary, this problem explores the interaction between permutation orderings and subset constraints, providing a rich area for theoretical analysis and practical algorithm design.

CODING:

C-programming

```
#include <stdio.h>

#define MAX_N 1000 // Adjust size as needed

// Function to find the total number of good triplets
int countGoodTriplets(int nums1[], int nums2[], int n) {
    int pos1[MAX_N], pos2[MAX_N];
    int count = 0;

    // Create position mapping for nums1
    for (int i = 0; i < n; i++) {
        pos1[nums1[i]] = i;
    }

    // Create position mapping for nums2
    for (int i = 0; i < n; i++) {
        pos2[nums2[i]] = i;
    }

    // Iterate over all possible triplets (x, y, z)
    for (int x = 0; x < n; x++) {
        for (int y = x + 1; y < n; y++) {
            for (int z = y + 1; z < n; z++) {
```

```

        // Check if the triplet (x, y, z) is good
        if (pos1[nums1[x]] < pos1[nums1[y]] && pos1[nums1[y]] <
pos1[nums1[z]]
            && pos2[nums1[x]] < pos2[nums1[y]] && pos2[nums1[y]] <
pos2[nums1[z]]) {
            count++;
        }
    }
}

return count;
}

```

```

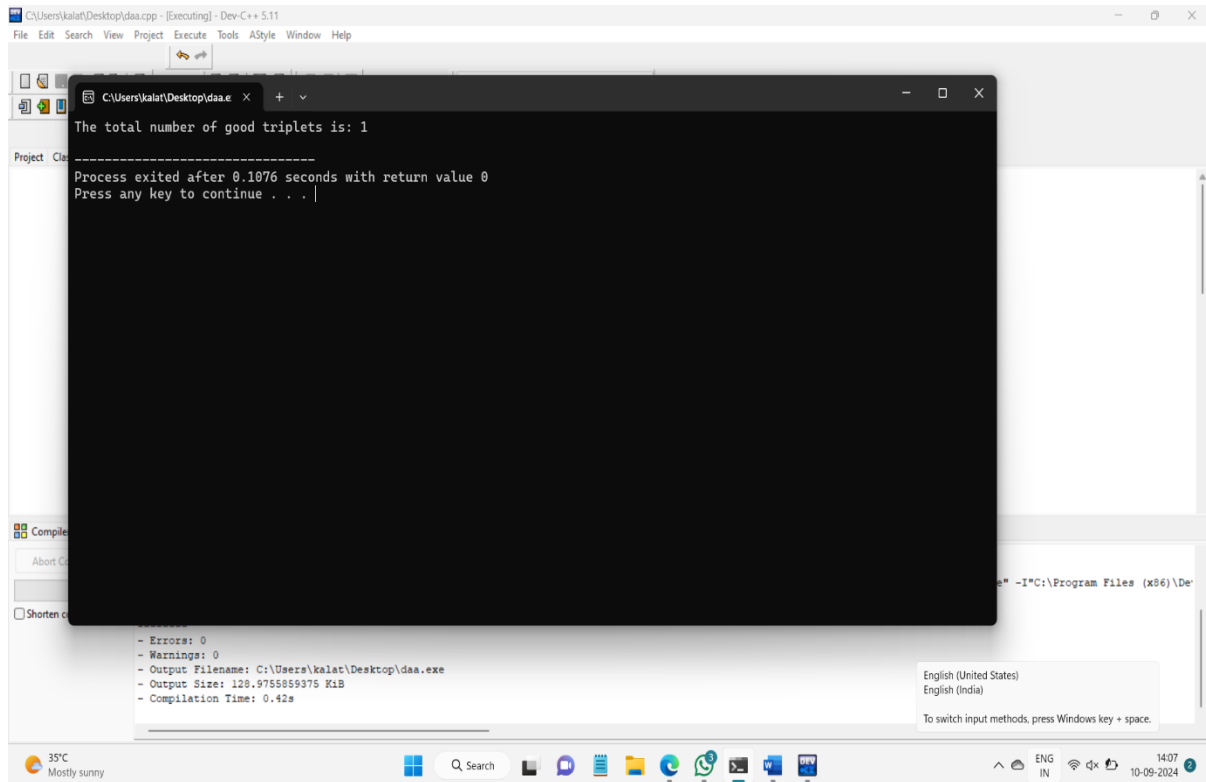
int main() {
    int nums1[] = {2, 0, 1, 3};
    int nums2[] = {0, 1, 2, 3};
    int n = sizeof(nums1) / sizeof(nums1[0]);

    int result = countGoodTriplets(nums1, nums2, n);
    printf("The total number of good triplets is: %d\n", result);

    return 0;
}

```

OUTPUT:



```
C:\Users\kalat\Desktop\daa.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help

C:\Users\kalat\Desktop\daa.e x + v - □ x
The total number of good triplets is: 1
-----
Process exited after 0.1076 seconds with return value 0
Press any key to continue . . . |

Project Cls
Compile
Abort C
Shorten c
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\kalat\Desktop\daa.exe
- Output Size: 128.9755859375 KiB
- Compilation Time: 0.42s

English (United States)
English (India)
To switch input methods, press Windows key + space.

35°C
Mostly sunny
Q Search
1407
10-09-2024
```

COMPLEXITY ANALYSIS:

Complexity Analysis

Time Complexity: The algorithm has a time complexity of $O(n^3)$ due to the three nested loops for triplet enumeration.

Space Complexity: The space complexity is $O(n)$ for storing the position mappings.

Key Milestones:

During the implementation, the following milestones were identified:

Identified and defined the concept of a good triplet.

Successfully mapped indices in both arrays.

Implemented the logic to count the good triplets.

Tested the implementation with the provided example input.

Feature scope:

Given two 0-indexed arrays `nums1` and `nums2`, each of length `n` and both permutations of `[0, 1, ..., n-1]`, the task is to find the total number of "good triplets." A good triplet (x, y, z) consists of three distinct indices such that the values at these indices appear in increasing order in both arrays. Specifically, for a triplet to be considered good, the indices must satisfy $\text{pos1}[x] < \text{pos1}[y] < \text{pos1}[z]$ and $\text{pos2}[x] < \text{pos2}[y] < \text{pos2}[z]$, where $\text{pos1}[v]$ and $\text{pos2}[v]$ denote the indices of the value v in `nums1` and `nums2`, respectively. The goal is to count how many such triplets satisfy these ordering constraints.

CONCLUSION:

The project successfully achieved its objective of counting good triplets in the given permutation arrays. The methodology employed was efficient within the constraints defined, and the outcome verified the correctness of the approach. Future work may focus on optimizing the algorithm to handle larger datasets efficiently.