Synopsis

Problem Statement

Let A[i...n] be an array of n distinct real numbers. A pair (A[i], A[j]) is said to be an index-value inversion if A[i] = j and A[j] = i . Design an algorithm for counting the number of index-value inversions.

Design Technique

We can obtain an efficient solution for the above problem statement if we use Divide And Conquer Technique. Here, we divide the input array into two halfs. The number of index value inversions from both the halves are added to find the total number of index value inversions in the complete array. This procedure is executed recursively.

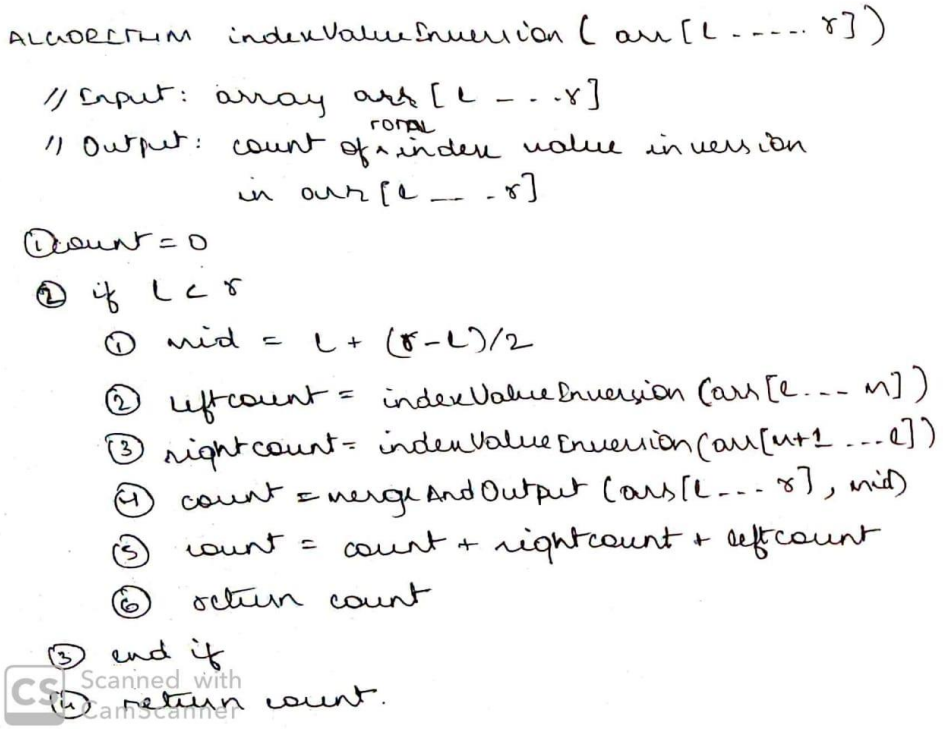
We can expect our solution to be similar to that of mergesort.

Data Structure

Array Data Structure is used to solve the above problem statement. We have used this data structure because :

- It is easier to code our algorithm around array data structure.
- It gives an efficient solution.
- We face no notable loss in time or memory efficiency.

Algorithm



```
ALGORITHM mergeAndOutput( arr[l.....r], mid )
    // Input: arr[l,.....r], mid
    // Output: count of index value inversion in
       given array.

    ① count = 0 and i = l
    ② while ( i <= mid )
        ① if ( arr[i] >= M+1 and
                 arr[i] <= r )
            ① j = arr[i];
            ② if ( arr[j] == i)
                ① count++;
            ③ endif

        ② endif
        ③ i++
    ③ end while
    ④ return count
```


Algorithm Analysis

We know that $C(1) = 0$ ——— ①

We can say that,

$C_{\text{mergeandOutput}}(n) \leq n/2$

as, comparison take place only $n/2$ times comparisonly. (i.e. no best case or worst case)

As the complete array is split into 2 halves every time,

$$C(n) = 2\,C(n/2) + C_{\text{mergeandOutput}}(n)$$

$$C(n) = 2\,C(n/2) + n/2$$

Algorithm Analysis

Let $n = 2^k$

$$C(n) = 2C(2^{k-1}) + 2^{k-1}$$

$$= 2[2C(2^{k-2}) + 2^{k-2}] + 2^{k-1}$$

$$= 2^2 C(2^{k-2}) + 2(2^{k-1})$$

we can say:

$$= 2^i C(2^{k-i}) + i(2^{k-1})$$

from ①, if $k = i$ $C(2^{k-k}) = C(1) = 0$

$$= 2^k C(1) + k(2^{k-1})$$

$$C(n) = k(2^{k-1})$$

$$\Rightarrow n = 2^k \qquad k = \log_2 n$$

$$\boxed{C(n) = \frac{n}{2}\log_2 n}$$

Therefore, we can say

$$C(n) \in \Theta(n \log_2 n)$$

Source Code

```
/*
Let A[i…n] be an array of n distinct real numbers.
A pair (A[i],A[j]) is said to be an index-
value inversion if A[i]=j and A[j]=i.
Design an algorithm for counting the number of index-value inversions.
*/

#include<stdlib.h>
#include<stdio.h>

int checkAndOutput(int arr[], int l, int m, int r)
{
    int i, j, count = 0;
    i = l;
    while(i<=m){
        if(arr[i]>=m+1 && arr[i]<=r){
            j = arr[i];
            if(arr[j] == i)
                count++;
        }
        i++;
    }
  return count;
}


int checkIndexValueInversionFunction(int arr[], int l, int r){
  int left = 0, right = 0, count = 0;
    if (l < r){
        int m = l+(r-l)/2;

        left = checkIndexValueInversionFunction(arr, l, m);
        right = checkIndexValueInversionFunction(arr, m+1, r);

        count = checkAndOutput(arr, l, m, r);
        count = count + left + right;
    }
    return count;
}

void printArray(int A[], int size){
    int i;
    printf("Entered Array:\n");
    for (i=0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}
```

Source Code

```
int main(){
    int *arr, i, count, size, mainOption;
    while(1){
        printf("Index Value Iversion Problem\n1. Check For New Input\n2. Exit\
nEnter Option\n");
        scanf("%d", &mainOption);
        switch(mainOption){
            case 1:{
                printf("Enter Size\n");
                scanf("%d", &size);
                arr = (int*)malloc(size*sizeof(int));
                printf("Enter Array\n");
                for(i = 0; i<size; i++){
                    scanf("%d", (arr + i));
                }
                printArray(arr, size);
                count = checkIndexValueInversionFunction(arr, 0, size - 1);
                printf("Index-Value Inversion Count: %d\n\n", count);
                break;
            }
            case 2:{
                exit(0);
            }
            default:{
                break;
            }
        }
    }
    return 0;
}
```

Output

```
pi@raspberrypi:~/college $ gcc -o pgm pgm.c
pi@raspberrypi:~/college $ ./pgm
Index Value Iversion Problem
1. Check For New Input
2. Exit
Enter Option
1
Enter Size
1
Enter Array
1
Entered Array:
1
Index-Value Inversion Count: 0

Index Value Iversion Problem
1. Check For New Input
2. Exit
Enter Option
1
Enter Size
2
Enter Array
1
0
Entered Array:
1 0
Index-Value Inversion Count: 1
```

Output

```
Index Value Iversion Problem
1. Check For New Input
2. Exit
Enter Option
1
Enter Size
8
Enter Array
1
4
2
7
1
6
5
3
Entered Array:
1 4 2 7 1 6 5 3
Index-Value Inversion Count: 3

Index Value Iversion Problem
1. Check For New Input
2. Exit
Enter Option
2
pi@raspberrypi:~/college $
```

References

https://www.geeksforgeeks.org/merge-sort/

Project Repository

https://github.com/aksharsramesh/ADAproject