Inversion

- Given a sequence of integers a_1, a_2,…, a_n. A pair (I, j) is called an inversion if $\boldsymbol{i < j}$ and $\boldsymbol{a_i > a_j}$
- Compute the number Q of inversions

- **Input**
  - Line 1: contains a positive integer n **(1 <= n <= 10^6)**
  - Line 2: contains a_1, a_2,…, a_n **(0 <= a_i <= 10^6)**
- **Output**
  - Write the value Q **module $10^9 + 7$**

- **Example**

stdin	stdout
6 3 2 4 5 6 1	6

Inversion: Hint

- Ý tưởng tương tự như sắp xếp hợp nhất (merge sort), chia mảng thành hai nửa (gần) bằng nhau trong mỗi bước cho đến khi đạt được trường hợp cơ sở (trường hợp khi chỉ có 1 phần tử trong mảng)

- Tạo một hàm đệ quy để chia mảng thành hai nửa.
  - Kết quả của mảng con từ left đến right là **tổng** số lần đảo ngược trong nửa đầu, số lần đảo ngược trong nửa sau và số lần đảo ngược khi hợp nhất cả hai.

- Tạo một hàm hợp nhất đếm số lần nghịch đảo khi hai nửa của mảng được hợp nhất
  - Nếu **a[i] > a[j],** thì có **(mid – i) nghịch đảo** (với a[i] thuộc nửa mảng thứ nhất, a[j] thuộc nửa mảng thứ 2, mid là phần tử cuối cùng của nửa thứ nhất)
  - Bởi vì mảng con bên trái và bên phải đã được sắp xếp, nên tất cả các phần tử còn lại trong mảng con bên trái (a[i+1], a[i+2] … a[mid]) sẽ lớn hơn a[j]

Implementation

```
#include <bits/stdc++.h>
#define maxn 1000006
using namespace std;
int const MOD = 1e9+7;
int n, a[maxn];
int temp[maxn];
```

Implementation

```
// Hàm hợp nhất 2 mảng và trả về số lượng cặp nghịch đảo khi hợp nhất.

int _merge(int left, int mid, int right)
{
    int i = left, j = mid + 1, k = left, inv_count = 0;
    while ((i <= mid) && (j <= right)) {
        if (a[i] <= a[j])   temp[k++] = a[i++];
        else {
            temp[k++] = a[j++];
            inv_count = (inv_count + (mid - i + 1)) % MOD;
        }
    }
    // Copy những phần tử còn lại của nửa bên trái (nếu còn) vào mảng trung gian
    while (i <= mid) temp[k++] = a[i++];
    // Copy những phần tử còn lại của nửa bên phải (nếu còn) vào mảng trung gian
    while (j <= right) temp[k++] = a[j++];
    // Copy những phần tử đã được hợp nhất vào mảng gốc
    for (i = left; i <= right; i++)     a[i] = temp[i];
    return inv_count;
}
```

Implementation

```
// An recursive function that sorts the input array and returns the number of inversions in the array.

int mergeSort (int left, int right){
    int mid, inv_count = 0;
    if (right > left) {
        // Divide the array into two parts and call mergeSort() for each of the parts
        mid = (right + left) / 2;
        // Inversion count will be sum of inversions in left-part, right-part and number of inversions in merging
        inv_count = (inv_count + mergeSort(left, mid)) % MOD;
        inv_count = (inv_count + mergeSort(mid + 1, right)) % MOD;
        inv_count = (inv_count + _merge(left, mid, right))% MOD;
    }
    return inv_count;
}

int main() {
    ios_base::sync_with_stdio(0); cin.tie(NULL); cout.tie(NULL);
    cin >> n;
    for (int i=1; i<=n; i++)     cin >> a[i];
    cout << mergeSort(1, n);
    return 0;
}
```