HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Applied Algorithm Lab

Inversions

ONE LOVE. ONE FUTURE.

Inversions

- Count the number of inversions (số cặp nghịch thế) of a permutation
- Input: a sequence of integers $a_1, a_2, \ldots, a_n$.
- Output: The number of pairs (i, j) that $i < j$ and $a_i > a_j$

- Example

stdin	stdout
6 3 2 4 5 6 1	6

Inversions

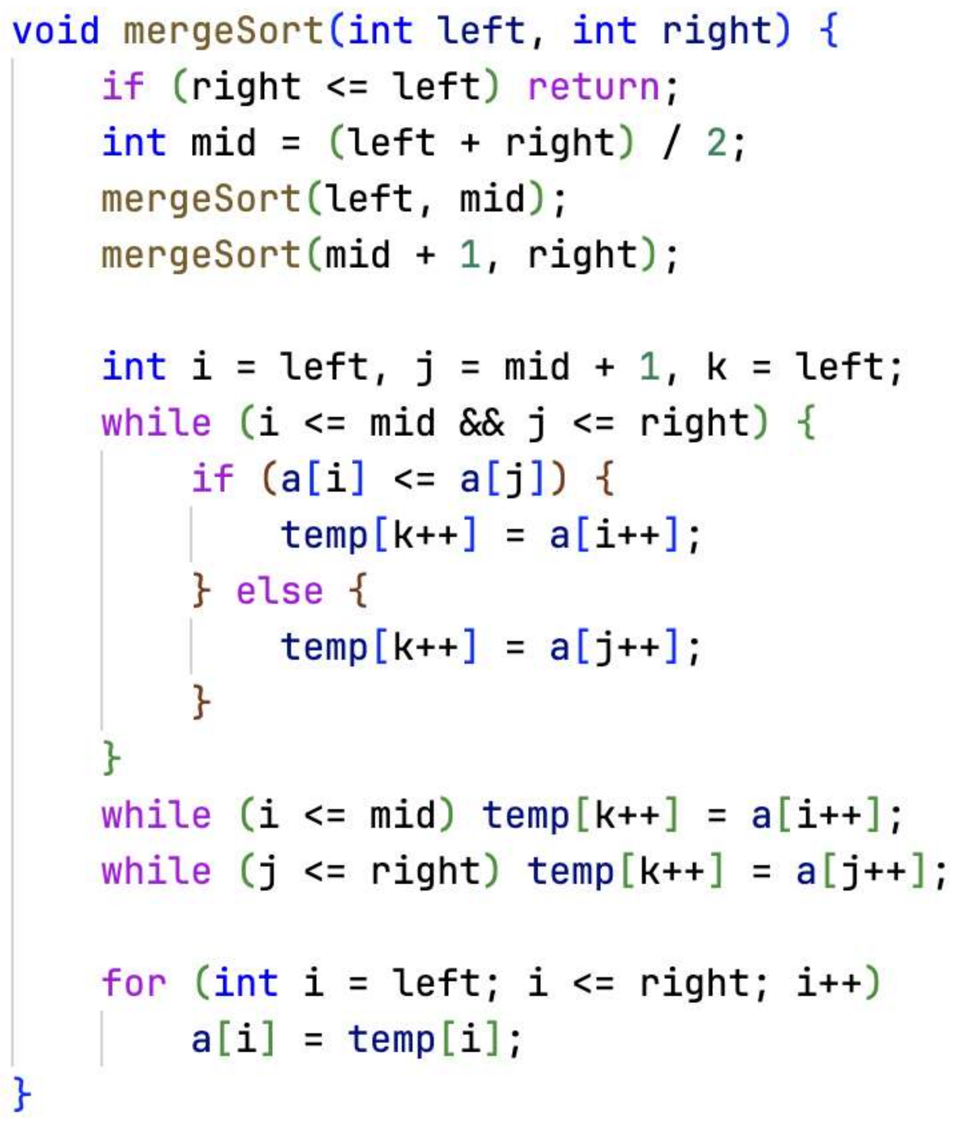
- Idea to solve #1: Brute-force from left -> right
  - At each element: count number of its right elements smaller than it -> number of inversions
  - Complexity: $O(n^2)$

Inversions

- Idea to solve #2: similar to merge sort
  - Fix the code mergeSort to sort and count inversions at the same time
  - Idea of mergeSort(int left, int right):
    - divide the array into 2 parts
    - Sort the subarray (left, mid) and subarray (mid +1, right)
    - Merge the two subarrays
    - Complexity: O(n log n)

Inversions

- Idea to solve #2: similar to merge sort
  - Code Merge sort:
  - Fix the code mergeSort to sort and count inversions at the same time



```
void mergeSort(int left, int right) {
    if (right <= left) return;
    int mid = (left + right) / 2;
    mergeSort(left, mid);
    mergeSort(mid + 1, right);

    int i = left, j = mid + 1, k = left;
    while (i <= mid && j <= right) {
        if (a[i] <= a[j]) {
            temp[k++] = a[i++];
        } else {
            temp[k++] = a[j++];
        }
    }
    while (i <= mid) temp[k++] = a[i++];
    while (j <= right) temp[k++] = a[j++];

    for (int i = left; i <= right; i++)
        a[i] = temp[i];
}
```

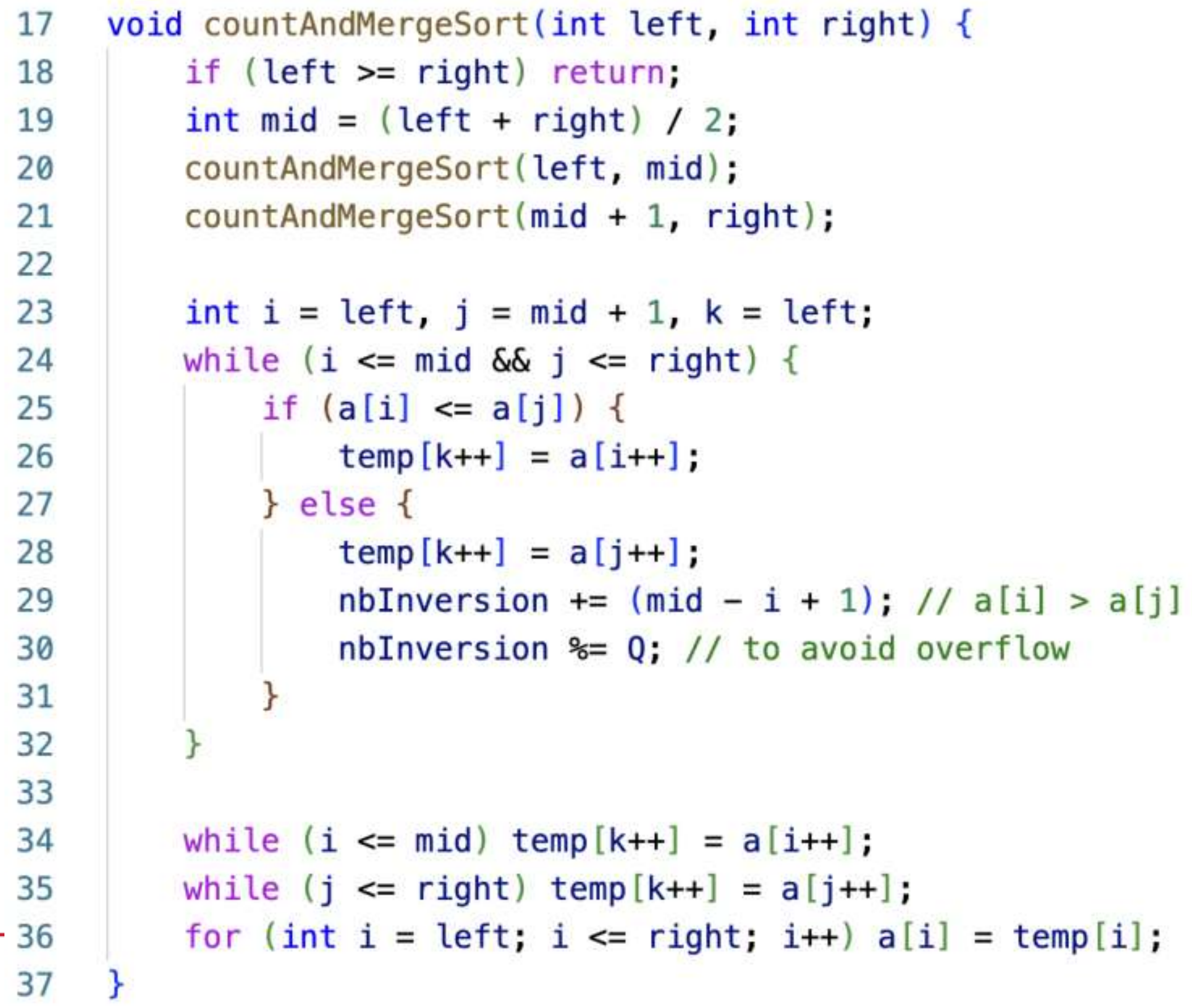
Inversions - Implementation

```
#include <bits/stdc++.h>
using namespace std;
#define N 1000001
const int Q = 1e9+7;

int n, a[N], temp[N], nbInversion = 0;

void input() {
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
}
```

Inversions - Implementation

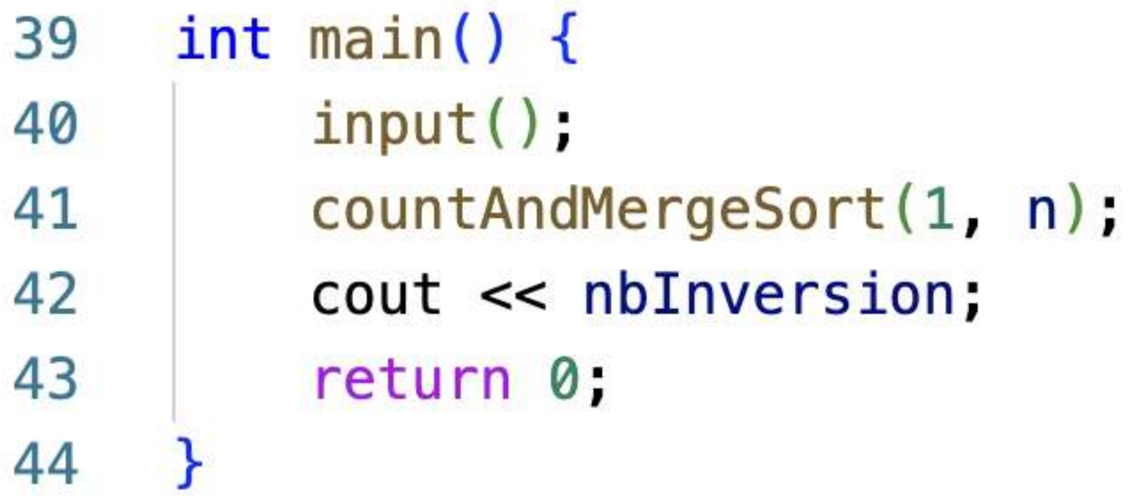


```
void countAndMergeSort(int left, int right) {
    if (left >= right) return;
    int mid = (left + right) / 2;
    countAndMergeSort(left, mid);
    countAndMergeSort(mid + 1, right);

    int i = left, j = mid + 1, k = left;
    while (i <= mid && j <= right) {
        if (a[i] <= a[j]) {
            temp[k++] = a[i++];
        } else {
            temp[k++] = a[j++];
            nbInversion += (mid - i + 1); // a[i] > a[j]
            nbInversion %= Q; // to avoid overflow
        }
    }

    while (i <= mid) temp[k++] = a[i++];
    while (j <= right) temp[k++] = a[j++];
    for (int i = left; i <= right; i++) a[i] = temp[i];
}
```

Inversions - Implementation



```cpp
int main() {
    input();
    countAndMergeSort(1, n);
    cout << nbInversion;
    return 0;
}
```

THANK YOU !