ADA23-HW1

許博翔

September 22, 2023

Problem 1. First, let's solve the following problem:

Given $\{a_i\}_{i=1}^n, \{b_i\}_{i=1}^n, \{c_i\}_{i=1}^n$, find $\displaystyle\sum_{(i,j) \text{ is an inversion in } \{a_i\}_{i=1}^n} b_i c_j$ in $O(n \log n)$ complexity.

Let $d_{l,r}(b,c) := \displaystyle\sum_{(i,j) \text{ is an inversion in } \{a_i\}_{i=l}^{r-1}} b_i c_j$.

Let's implement $solve(l,r)$ such that it does the following things:

1. Sort $\{a_i\}_{i=l}^{r-1}, \{b_i\}_{i=l}^{r-1}, \{c_i\}_{i=l}^{r-1}$ by the order of $\{a_i\}_{i=l}^{r-1}$. (in other words, sort $\{(a_i, b_i, c_i)\}_{i=l}^{r-1}$ by a_i).
2. Return $d_{l,r}(b,c)$.

Use divide and conquer to implement it.

For the base case $r \le l+1$, just do nothing and return $d_{l,r}(b,c) = 0$.

For the other case $r \ge l+2$, let $m := \lfloor \frac{l+r}{2} \rfloor$.

First, do $solve(l,m)$ and $solve(m,r)$.

There are 3 kinds of inversions (i,j):

1. $i < j < m$, the summation of $b_i c_j$ of this kind of inversions is exactly $d_{l,m}(b,c)$, which is counted by $solve(l,m)$.
2. $m \le i < j$, the summation of $b_i c_j$ of this kind of inversions is exactly $d_{m,r}(b,c)$, which is counted by $solve(m,r)$.
3. $i < m \le j$.

Since $\{a_i\}_{i=l}^{m-1}, \{a_i\}_{i=m}^{r-1}$ have been sorted by $solve(l, m), solve(m, r)$, respectively, we can do the merge part in the merge sort to sort $\{a_i\}_{i=l}^{r-1}, \{b_i\}_{i=l}^{r-1}, \{c_i\}_{i=l}^{r-1}$ by $\{a_i\}_{i=l}^{r-1}$ in $O(r-l)$ time complexity.

Set C to 0 and $d_{l,r}(b,c)$ to $d_{l,m}(b,c) + d_{m,r}(b,c)$.

Do the following when merging $L := \{a_i\}_{i=l}^{m-1}, R := \{a_i\}_{i=m}^{r-1}$ to the sorted array A:

1. If we put an element a_i of R to A, increase C by c_i.
2. If we put an element a_i of L to A, increase $d_{l,r}(b,c)$ by $b_i C$.

Note that for the tie breaker, we put the element in L instead of that in R to A, so that whenever an element a_i of L is put into A, $a_i >$ any element a_j in A that are from R, $a_i \leq$ any element a_j that are not in A, and therefore (i, j) forms an inversion of the third kind if and only if a_j is in A and is from R.

Since in 1. we maintain $C = \sum_{a_i \text{is from } R \text{ and is in } A} c_i$, we'll increase $d_{l,r}(b,c)$ by $\sum_{(i,j) \text{ is an inversion and } j \geq m} b_i c_j$ in 2.

$\therefore$ after merging L, R, the arrays are sorted, and we finish counting $d_{l,r}(b,c)$.

Since the time complexity for a single 1. or 2. is $O(1)$, and there are $O(r-l)$ elements to be merged, the time complexity of the merging part is $O(r-l)$.

Let $T(r-l)$ denote the time of $solve(l, r)$.

The time complexity of the dividing part is $2T((r-l)/2)$, of the merging part is $O(r-l)$.

$\Rightarrow T(r-l) = 2T((r-l)/2) + O(r-l)$.

By the master theorem, $T(r-l) = O((r-l)\log(r-l))$.

$\therefore T(n) = O(n \log n)$.

Back to (a), (b), (c):

(a) is $d_{l,r}(b,c)$, where $b_i := c_i := 1$, which can be solved in $O(n \log n)$ time complexity.

Trivially, (b) can be solved if (c) is solved.

(c) is $\sum_{i=0}^{k} \binom{k}{i} d_{l,r}(b^{(i)}, c^{(k-i)})$ by the binomial theorem, where $b_j^{(i)} := c_j^{(i)} := a_j^i$.

Since $\binom{k}{0} = 1$, $\forall i$, $\binom{k}{i+1} = \binom{k}{i} \cdot \frac{k-i}{i+1}$.

$\therefore \binom{k}{0}, \binom{k}{1}, \ldots, \binom{k}{k}$ can be counted in $O(k)$ time complexity.

Since each $d_{l,r}(b^{(i)}, c^{(k-i)})$ can be counted in $O(n \log n)$ time complexity, the total time complexity of (c) is $O(nk \log n + k) = O(nk \log n)$.

Problem 2.