ADA23-HW1 Solution

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):

K-(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 $solve(l,m)$.

K-(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 $solve(m,r)$.

K-(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:

M-(1) If we put an element a_i of R to A, increase C by c_i.

M-(2) If we put an element a_i of L to A, increase $d_{l,r}(b, c)$ by b_iC.

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

$\therefore$ after merging L, R, the arrays are sorted, and we finish counting the summation of $b_i c_j$ of K-(3) (i, j).

Return $d_{l,r}(b, c)$.

Since the time complexity for a single M-(1) or M-(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 $\displaystyle\sum_{i=0}^{k} \binom{k}{i} d_{l,r}(b^{(i)}, c^{(k-i)})$ by the binomial theorem, where $b_j^{(i)} := c_j^{(i)} := a_j^i$.

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

$\therefore \binom{k}{0}, \binom{k}{1}, \ldots, \binom{k}{k}$ 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.

(d)

Let $m := \lfloor \frac{n}{3} \rfloor$.

Construction:

For $1 \le i \le n-m$, the i-th set operation is to insert i.

For $n-m+1 \le i \le n$, the i-th set operation is to delete $n-i+1$.

The number of stack operations:

In the first $n-m$ set operations, each contains one push operaion.

In the last m set operations, the i-th one is to delete $n-i+1$, and before it, all delete operations are to delete $m, m-1, \ldots, n-i+2$. Since the position of $n-i+1$ is under those of $m, m-1, \ldots, n-i+2$, when deleting $m, m-1, \ldots, n-i+2$, the position of $n-i+1$ won't be changed in Arctan's implementation, which means it will be under the position of $m+1, m+2, \ldots, n$. Therefore, to delete $n-i+1$, Arctan needs to pop $m+1, m+2, \ldots, n$ first, then pop $n-i+1$, finally push $m+1, m+2, \ldots, n$ back to the stack, which takes $2(n-m)+1$ stack operations in total.

$\therefore$ the number of stack operations in total is $(n-m) + m(2(n-m)+1) = n - m + 2nm - 2m^2 + m = n + 2\lceil \frac{2n}{3} \rceil \lfloor \frac{n}{3} \rfloor = \Theta(n^2)$.

(e)

Define $B_{l,r}$ as $\left(\bigcup_{i=l}^{r-1} A_i \setminus A_{i+1} \right) \setminus \left(\bigcup_{i=l}^{r-1} A_{i+1} \setminus A_i \right)$. That is, the set of all elements that will be deleted but not be inserted during the l-th to the $r-1$-th set operation.

Define $C_{l,r}$ as $\left(\bigcup_{i=l}^{r-1} A_{i+1} \setminus A_i \right) \setminus \left(\bigcup_{i=l}^{r-1} A_i \setminus A_{i+1} \right)$. That is, the set of all elements that will be inserted but not be deleted during the l-th to the $r-1$-th set operation.

Define $S_{i,j}$ as $\begin{cases} \text{the } j\text{-th element counted from the bottom of the stack } S_i, \text{ if } j > 0; \\ \text{the } (-j)\text{-th element counted from the top of the stack } S_i, \text{ if } j < 0. \end{cases}$.

Define $S_{i,l..r}$ as the stack containing $r-l+1$ elements, where $\forall 1 \le j \le r-l+1$, the j-th element counting from the bottom is $S_{i,l+j-1}$.

Define A_S as the set of the elements of a stack S.

Define $S + a$ as the stack formed by pushing an element a into a stack S.

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

C-(1) Given is a stack S_L such that $A_{S_L} = A_l$ and $A_{S_{L,-|B_{l,r}|..-1}} = B_{l,r}$.

C-(2) Return is a stack S_R such that $A_{S_R} = A_r$ and $S_{R,1..|S_L|-|B_{l,r}|} = S_{L,1..|S_L|-|B_{l,r}|}$.

C-(3) $\exists i_l, i_{l+1}, \ldots, i_r$ where $L = i_l < i_{l+1} < \cdots < i_r = R$ such that $\forall l \le j \le r,\ A_{S_{i_j}} = A_j$.

Use divide and conquer to implement it.

For the base case $r \le l+1$, if $A_{l+1} = A_l \cup \{a\}$ for some a, then let $S_R = S_{L+1} = S_L + a$; otherwise let $S_R = S_{L+1} = S_{L,1..|S_L|-1}$ since by C-(1), $\{S_{L,-1}\} = A_l \setminus A_{l+1}$. One can easily check that C-(2) and C-(3) are satisfied.

For the other case $r \ge l+2$, let $k := \lfloor \frac{l+r}{2} \rfloor$, and do the following:

O-(1) Pop the top $|B_{l,r}|$ elements from the stack, and the resulting stack is $S_{L+|B_{l,r}|}$.

O-(2) Push all the elements in $B_{l,r} \setminus B_{l,k}$ into the stack, then push all the elements in $B_{l,k}$ into the stack, and the resulting stack is S_{L_1}, where $L_1 := L + 2|B_{l,r}|$.

O-(3) Do $solve(l, k)$. Since O-(1), O-(2) make $A_{S_{L_1}} = A_{S_L} = S_l$, and O-(2) gaurantees that $A_{S_{L_1,-|B_{l,k}|..-1}} = B_{l,k}$, C-(1) is satisfied. Suppose the returning stack is S_{R_1}.

O-(4) Let $D := B_{l,r} \setminus B_{l,k} \cup C_{l,k}$. Pop the top $|D|$ elements from the stack, and the resulting stack is $S_{R_1+|D|}$.

O-(5) Push all the elements in $D \setminus B_{k,r}$ into the stack, then push all the elements in $B_{k,r}$ into the stack, and the resulting stack is S_{L_2}, where $L_2 := R_1 + 2|D|$.

O-(6) Do $solve(k, r)$. Since O-(4), O-(5) make $A_{S_{L_2}} = A_{S_{R_1}} \overset{\text{C-(2)}}{=} A_k$, and O-(5) gaurantees that $A_{S_{L_2,-|B_{k,r}|..-1}} = B_{k,r}$, C-(1) is satisfied. Suppose the returning stack is S_{R_2}.

O-(7) Let $R := R_2$, return S_R. Since by C-(2), $A_{S_R} = A_{S_{R_2}} = A_r$, and none of the above changes the $|S_L| - |B_{l,r}|$ elements in the bottom, C-(2) is satisfied. Since by C-(3), $solve(l,k), solve(k,r)$ gaurantee the existense of $i_l, i_{l+1}, \ldots, i_r$, C-(3) is satisfied.

Let $T(r-l)$ denote the number of stack operations in $solve(l,r)$.
For O-(1), (2), (4), (5), there are $2|B_{l,r}|+2|B_{l,r}|-2|B_{l,k}|+2|C_{l,k}|$ of stack operations in total. Since $B_{l,r}, B_{l,k}, C_{l,k} \leq r-l$, $2|B_{l,r}|+2|B_{l,r}|-2|B_{l,k}|+2|C_{l,k}| = O(r-l)$.
The number of stack operations in (3), (6) is $T((r-l)/2)$.
$\Rightarrow T(r-l) = 2T((r-l)/2) + O(r-l)$.
By the master theorem, $T(r-l) = O((r-l)\log(r-l))$.
$\therefore m = T(n) = O(n \log n)$.