ESO207A
Assignment 1

Rahul Sethi
190668

Q1 (a)

Algorithm 1 Count Inversions - Divide and Conquer

Input: An array A of size n
Output: Sorted (ascending) version of array A

```
B[i] ⟵ 0 for all 1 ≤ i ≤ n
return InversionCount(A, B, 1, n)
function InversionCount(A, B, start, end)
    count ⟵ 0
    if start < end then
        mid ⟵ ⌊(start + end)/2⌋
        count += InversionCount(A, B, start, mid)          ▷ count inversions in left subarray
        count += InversionCount(A, B, mid + 1, end)        ▷ count inversions in right subarray
        count += MergeAndCount(A, B, start, mid, end)      ▷ count cross-inversions
    end if
    return count
end function
function MergeAndCount(A, B, start, mid, end)
    p ⟵ start
    q ⟵ mid + 1
    r ⟵ start
    count ⟵ 0
    while p ≤ mid and q ≤ end do
        if A[p] ≤ A[q] then
            B[r] ⟵ A[p]
            r += 1
            p += 1
        else
            B[r] ⟵ A[q]
            count += mid − p + 1
            r += 1
            q += 1
        end if
    end while
```

Continued on next page

[0]sethir@iitk.ac.in, 190668, BS MTH

```
30:     while p ≤ mid do
31:         B[r] ⟵ A[p]
32:         r += 1
33:         p += 1
34:     end while
35:     while q ≤ end do
36:         B[r] ⟵ A[q]
37:         r += 1
38:         q += 1
39:     end while
40:     while start ≤ i ≤ end do                    ▷ copy sorted sequence to original array
41:         A[i] ⟵ B[i]
42:         i += 1
43:     end while
44:     return count
45: end function
```

Q1 (b)

Definition 1

Given an array $A[start : end]$, a pair (i, j) for $start \leq i < j \leq end$ is an *inversion* if $A[i] > A[j]$.

Definition 2

Given $start \leq p < q \leq end$, the boolean $S(p, q)$ is true if $A[p] > A[q]$, and false otherwise.

Details and Correctness of InversionCount:

If $start \geq end$, the number of inversions $= 0$ (one or no elements in the array). Otherwise, divide the array into two parts, $A[start : mid]$ and $A[mid + 1 : end]$, where $mid = \lfloor (start + end)/2 \rfloor$. Any inversion (i, j) in $A[start : end]$ is of one of the three following types:

1. $i < j \leq mid$, which implies (i, j) is an inversion in $A[start : mid]$. Let such inversions belong to the set I_1. (Line 7)
2. $mid + 1 \leq i < j$, which implies (i, j) is an inversion in $A[mid + 1 : end]$. Let such inversions belong to the set I_2. (Line 8)
3. $i \leq mid < j$ - these are *cross inversions*, let them belong to the set C. (Line 9)

Clearly, we need to return $|I_1| + |I_2| + |C|$. Suppose we have computed (recursively) the number of inversions in these two arrays as $|I_1|$ and $|I_2|$ respectively. The MergeAndCount function (line 13) merges the two sorted[1] sub-arrays, while counting the cross-inversions.

Details and Correctness of MergeAndCount:

Two pointers p and q are initialized to $start$ and $mid + 1$ respectively (Lines 14-15), which are incremented within the respective sub-arrays as the algorithm proceeds. Moreover, $A[start : mid]$ and $A[mid + 1 : end]$ are sorted. The array B is used to collect elements from $A[start : mid]$ and

[1] If $A[start : mid]$ and $A[mid + 1 : end]$ are sorted, the number of cross inversions can be calculated in $O(n)$ time, where $n = end - start + 1$

$A[mid + 1 : end]$ such that $B[start : end]$ is sorted. (Lines 20, 24, 31, 36). B's elements are later copied onto A in Lines 40-43.

Lemma A

The number of cross-inversions between $A[start : mid]$ and $A[mid + 1 : end]$ is the same as that between sorted($A[start : mid]$) and sorted($A[mid + 1 : end]$).[2]

Proof

Let (i, j) be a cross-inversion, i.e. $i \leq mid < j$ and $A[i] > A[j]$. After sorting $A[start : mid]$ and $A[mid + 1 : end]$ separately, let $i \mapsto i'$ and $j \mapsto j'$. $A[i'] > A[j']$ holds. Moreover, $i' \leq mid$ and $j' > mid$. This implies $i' \leq j'$ and $A[i'] > A[j']$, i.e. (i', j') is a cross-inversion.

$\blacksquare$

Lemma B

For some $q = q'$, let $S(p, q)$ be true for the first time when $p = p'$, i.e. $A[p'] > A[q']$. Then, $I_C(q') = \{(p^*, q') : p^* \geq p'\}$ is the set of all cross-inversions involving q'.

Proof

Since $A[start : mid]$ is sorted, $A[p^*] > Q[j'] \ \forall p^* \geq p'$, and so all such (p^*, q') are inversions. To show that this is the complete set of inversions involving q', consider some $p'' < p'$. Since the pointer p is currently at $p' > p''$, at some earlier stage the algorithm must have compared $A[p'']$ and $A[q'']$, where $q'' \leq q'$ - and found that $A[p''] \leq A[q'']$, i.e. $S(p'', q'')$ was false. Since $A[mid + 1 : end]$ is sorted, this implies $A[p''] \leq A[q']$. As a result, $\forall p'' < p'$, (p'', q') is not an inversion. Hence, $I_C(q') = \{(p^*, q') : p^* \geq p'\}$ is the set of all cross-inversions involving q'.[3]

$\blacksquare$

Correctness of Cross-Inversion Count

Consider the while loop in MergeAndCount (Lines 18-29). If $S(p, q)$ is false, *count* is not incremented. Else, if $S(p, q)$ is true, *count* is incremented by $mid - p + 1$ (follows from Lemma B), which exactly corresponds to $|I_C(q)|$ defined earlier.

Correctness of Merge

Consider the while loops in MergeAndCount (Lines 18-39). We define the following loop invariants:[4]

1. ϕ_1: $B[start : r - 1]$ is sorted.
2. ϕ_2: $B[start : r - 1]$ is a permutation of $A[start : p - 1] \cup A[mid + 1 : q - 1]$[5]
3. ϕ_3:
   (a) $B[start : r - 1] \leq A[p], A[q]$ if $p \leq mid$ and $q \leq end$

[2] sorted($A[l : l']$) refers to the permutation of $A[l : l']$ wherein $A[i] \leq A[j] \ \forall l \leq i < j \leq l'$.

[3] Note that $C = \bigcup_{q'=mid+1}^{end} I_C(q')$

[4] In particular, we show that (i) $B[start : end]$ is sorted after completion of MergeAndCount, i.e. $A[start : mid]$ and $A[mid + 1 : end]$ are merged correctly, and (ii) $count$ = number of inversions *seen so far*

[5] Slight abuse of notation. This does not account for repeated elements. However, the generalization is fairly straightforward, and a sketch of the same has been provided in the appendix.

(b) $B[start : r-1] \leq A[p]$ if $p \leq mid$ and $q > end$

(c) $B[start : r-1] \leq A[q]$ if $p > mid$ and $q \leq end$

4. ϕ_4: $start \leq p \leq mid+1$, $mid+1 \leq q \leq end+1$, and $r = p+q-mid-1$
5. ϕ_5: $count$ = number of *cross-inversions* corresponding to the indices in the interval $\mathcal{Q}(q) = [mid+1, q)$[6]

Invariance of ϕ_1 -
Before entering the while loop (Line 18), $r = start$ and $B[start : r-1]$ is empty, hence the claim ϕ_1 is vacuously true. Assume ϕ_1 holds before a particular iteration of the loop, when $(p, q, r) = (p', q', r')$.

- If $S(p, q)$ is false, $B[r'] = A[p']$ and $(p, r) = (p'+1, r'+1)$. We show that $B[r'-1] \leq B[r']$:
  - If $B[r'-1] = A[p'-1]$, then $B[r'-1] \leq A[p'] = B[r']$ as $A[start : mid]$ is sorted.
  - If $B[r'-1] = A[q'-1]$, then $S(p', q'-1)$ is true and $A[p'] > A[q'-1]$. As a result, $B[r'-1] < A[p'] = B[r']$.
- If $S(p, q)$ is true, $B[r'] = A[q']$ and $(q, r) = (q'+1, r'+1)$. We show that $B[r'-1] \leq B[r']$:
  - If $B[r'-1] = A[q'-1]$, then $B[r'-1] \leq A[q'] = B[r']$ as $A[mid+1 : end]$ is sorted.
  - If $B[r'-1] = A[p'-1]$, then $S(p'-1, q')$ is false and $A[p'-1] \leq A[q']$. As a result, $B[r'-1] \leq A[q'] = B[r']$.

As $B[r'-1] \leq B[r']$, $B[start : r']$ is sorted, ϕ_1 continues to hold after the iteration as well. Note that when $p = mid+1$ or $q = end+1$ (one of the two sorted sub-arrays exhausted), we exit the loop on Lines 18-29 and simply copy the elements of the remaining array onto B (Lines 30-39). Finally $(p, q, r) = (mid+1, end+1, end+1)$, and $B[start : end]$ is sorted.

■

Invariance of ϕ_2 -
Before entering the while loop (Line 18), $r = start$ and $B[start : r-1]$ is empty, hence the claim ϕ_2 is vacuously true. Assume ϕ_2 holds before a particular iteration of the loop, when $(p, q, r) = (p', q', r')$.

- If $S(p, q)$ is false, $B[r'] = A[p']$ and $(p, r) = (p'+1, r'+1)$. Clearly, $B[start : r']$ is a permutation of $A[start : p'] \cup A[mid+1 : q'-1]$.
- If $S(p, q)$ is true, $B[r'] = A[q']$ and $(q, r) = (q'+1, r'+1)$. Clearly, $B[start : r']$ is a permutation of $A[start : p'-1] \cup A[mid+1 : q']$.

ϕ_2 continues to hold after the iteration as well. Finally $(p, q, r) = (mid+1, end+1, end+1)$ and $B[start : end]$ is a permutation of $A[start : mid] \cup A[mid+1 : end]$, as required.

■

Invariance of ϕ_3 -
Before entering the while loop (Line 18), $r = start$ and $B[start : r-1]$ is empty, hence the claim ϕ_3 is vacuously true. Assume ϕ_3 holds before a particular iteration of the loop, when $(p, q, r) = (p', q', r')$. We shall prove the three cases separately:

1. $B[start : r'-1] \leq A[p'], A[q']$ if $p' \leq mid$ and $q' \leq end$ holds to begin with.
   - If $S(p, q)$ is false, $B[r'] = A[p']$ and $(p, r) = (p'+1, r'+1)$. Since $A[start : mid]$ is sorted, $B[r'] = A[p'] \leq A[p'+1]$. So, $B[start : r'] \leq A[p'+1], A[q']$ holds.

[6] When $q = end+1$, we're done.

- If $S(p,q)$ is true, $B[r'] = A[q']$ and $(q,r) = (q'+1, r'+1)$. Since $A[mid+1 : end]$ is sorted, $B[r'] = A[q'] \leq A[q'+1]$. So, $B[start : r'] \leq A[p'], A[q'+1]$ holds.

2. $B[start : r'-1] \leq A[p']$ if $p' \leq mid$ and $q' > end$ holds to begin with (loop in Lines 30-34 is being executed). After the iteration, $(p,r) = (p'+1, r'+1)$ and $B[r'] = A[p'] \leq A[p'+1]$ as $A[start : mid]$ is sorted. So, $B[start : r'] \leq A[p'+1]$ holds.

3. $B[start : r-1] \leq A[q]$ if $p > mid$ and $q \leq end$ holds to begin with (loop in Lines 35-39 is being executed). After the iteration, $(q,r) = (q'+1, r'+1)$ and $B[r'] = A[q'] \leq A[q'+1]$ as $A[mid+1 : end]$ is sorted. So, $B[start : r'] \leq A[q'+1]$ holds.

ϕ_3 continues to hold after the iteration as well.

$\blacksquare$

Invariance of ϕ_4 -
Before entering the while loop (Line 18), $(p,q,r) = (start, mid+1, start)$. Also, $p+q-mid-1 = start = r$. So, $start \leq p \leq mid+1$, $mid+1 \leq q \leq end+1$, and $r = p+q-mid-1$ are clearly true. Assume ϕ_4 holds before a particular iteration of the loop, when $(p,q,r) = (p',q',r')$. After the iteration, we have one of the following cases:

- $(p,r) = (p'+1, r'+1)$
  $r' = p'+q'-mid-1 \iff (r'+1) = (p'+1)+q'-mid-1$. Additionally, the loop condition guarantees that $p' \leq mid$, so that $p'+1 \leq mid+1$. $start \leq p'+1 \leq mid+1$ is satisfied.
- $(q,r) = (q'+1, r'+1)$
  $r' = p'+q'-mid-1 \iff (r'+1) = p'+(q'+1)-mid-1$. Additionally, the loop condition guarantees that $q' \leq end$, so that $q'+1 \leq end+1$. $mid+1 \leq q'+1 \leq end+1$ is satisfied.

The operations on (p,q,r) in Lines 30-39 are a proper subset of those in lines 18-29, so the above proof suffices. ϕ_4 continues to hold after the iteration as well.

$\blacksquare$

Invariance of ϕ_5 - Before entering the while loop (Line 18), $(p,q,r) = (start, mid+1, start)$. The interval $\mathcal{Q}(q)$ is empty, and $count = 0$. Assume ϕ_5 holds before a particular iteration of the loop, when $(p,q,r) = (p',q',r')$. After the iteration, we have one of the following cases:

- If $S(p,q)$ is false, q is unchanged. ϕ_5 continues to hold.
- If $S(p,q)$ is true, $q = q'+1$ and $count$ is incremented by $mid-p+1$. Correctness of this step, and hence the invariance of ϕ_5 follow from Lemma B proved earlier.

ϕ_5 continues to hold after the iteration as well. Lastly, when $q = end+1$, $count$ = number of *cross-inversions* corresponding to the indices in the interval $\mathcal{Q}(q) = [mid+1, end+1)$, as desired.
This completes the proof.

$\blacksquare$

Time Complexity Analysis

Consider Lines 7-9 in the function InversionCount. The input array is of size $n = end - start + 1$. Clearly, the size of the left subarray $\lceil \frac{n}{2} \rceil$ and that of the right subarray is $\lfloor \frac{n}{2} \rfloor$. Let us first prove a lemma, which will be helpful in deducing the time complexity of this algorithm.

Lemma C:
If $T(n)$ satisfies the following recurrence, then $T(n) \leq n\lceil \log n \rceil \ \forall n \in \mathbb{N}$ (all logarithms have base 2)

$$T(n) \le \begin{cases} 0 & \text{if } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n & \text{if } n > 1 \end{cases}$$

Proof of Lemma C:
(Strong induction on n)
The base case ($n = 1$) is satisfied. Define $a = \lfloor \frac{n}{2} \rfloor$ and $b = \lceil \frac{n}{2} \rceil$ (notice that $a + b = n$). Also, it is worth noting that $\log b \le \lceil \log n \rceil - 1$ (proof shown below)

$$\begin{aligned} b &= \lceil n/2 \rceil \\ &\le \lceil 2^{\lceil \log n \rceil - 1} \rceil \\ &= 2^{\lceil \log n \rceil - 1} \end{aligned}$$

$$\log b \le \lceil \log n \rceil - 1$$

Induction Step: Assume Lemma C holds for $1, 2, ..., n-1$.

$$\begin{aligned} T(n) &\le T(a) + T(b) + n \\ &\le a\lceil \log a \rceil + b\lceil \log b \rceil + n \\ &\le a\lceil \log b \rceil + b\lceil \log b \rceil + n \\ &= n\lceil \log b \rceil + n \\ &\le n\lceil (\lceil \log n \rceil - 1) \rceil + n \\ &= n(\lceil \log n \rceil - 1) + n \\ &= n \log n \end{aligned}$$

■

Proposition:
The InversionCount algorithm counts the number of inversions in a permutation of size n in $O(n \log n)$ time.

Proof:
The worst case running time $T(n)$ satisfies the recurrence:

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + O(n) & \text{if } n > 1 \end{cases}$$

The above piece-wise definition of $T(n)$ is motivated by the following observations:

- The problem is *divided* into two halves (Lines 7-8) of sizes $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$.
- *Combining* solutions to the two halves takes $O(n)$ time - as p and q visit all elements in $A[start : mid]$ and $A[mid + 1 : end]$ exactly once. All other operations take constant time.

The solution of the above-mentioned recurrence, and hence the proof of this proposition follow from Lemma C.

■

Appendix

1. About Notation in ϕ_2

ϕ_2 is defined as - "$B[start : r-1]$ is a permutation of $A[start : p-1] \cup A[mid+1 : q-1]$".
As set notation prohibits us from working with multiple instances of the same element, the loop invariant ϕ_2 really only works for arrays with all distinct elements. However, we can easily get around this problem by defining a **new ordering** over the elements of the array, as follows:

1. If $A[i] > A[j]$, then $A[i]$ is **greater than** $A[j]$.
2. If $A[i] < A[j]$, then $A[i]$ is **less than** $A[j]$.
3. If $A[i] = A[j]$, then:
   - If $i = j$, then $A[i]$ is **equal to** $A[j]$.
   - If $i < j$, then $A[i]$ is **less than** $A[j]$.
   - If $i > j$, then $A[i]$ is **greater than** $A[j]$.

It is easy to see that this *new ordering* is in fact *consistent*.