Date
Page

A.1 Merge sort (A, i, j)

```
mid = (i+j)/2 , count = 0
  if j > i then
        count += mergesort(A, i, m)
        count += mergesort(A, m+1, j)
        Count += merge sort(A, i, m+1, j)
  return count.
// end func.
// Func Start.
   Merge (A, i, m, j)
   Count = 0 , ~~left = i, p=i, q=m~~
     left = i , mid = m , k = i

while( left ≤ j-1 & mid <= j)
     if ( A[left] <= A[mid] )
              temp [A] = A [left]
                 k = k+1 , left = left+1
     else  & temp [k] = A [mid]
                 k = k+1, mid = mid+1, count = count + j
                                                  - left
while(left <= j-1)
         temp [k] = A [left]
              k = k+1, left = left+1
while ( mid <= j)
              temp [k] = A [mid]
                 k = k+1 , mid = mid+1
while ( left < length of array)
       A [left] = temp [left]
                 left ++
   return count
// end merge.
```

In main func." pass array to mergesort

Q.2 Correctness of while loop in function "merge"

→ We have 2 arrays separately sorted.
→ We have to return the variable 'count', which is our result.

Loop invariants

→ ① $b[i..r-1]$ is sorted and is a permutation of $a[i...p-1] \cup a[mid+1 .. q-1]$
→ ② $b[i .. r-1] \leq a[p], a[q]$ if $p \leq mid$, $q \leq j$
$b[i..r-1] \leq a[q]$ if $p > mid$, $q <= j$
$b[i..r-1] <= a[p]$ if $p <= mid$, $q > j$
→ ③ $i <= p \leq mid+1$ $mid+1 <= q \leq j+1$, $r = p+q-mid-1$
→ ④ count is number of inversion of form (l, k)
$a[l] > a[k]$ $i <= l <= mid$ & $mid+1 <= k <= q-1$

Initially

(a) $b[i..r-1]$ is empty
(b) $p = i \Rightarrow$ $i <= p <= mid+1$
$q = mid+1 \Rightarrow mid+1 <= q <= j+1$
$r = i$, $p = i$, $q = mid+1 \Rightarrow r = p+q-mid-1$
(c) $q = mid+1$ so our k is undefined. So $count = 0$

Let value of p be p_0, $q \to q_0$, $r \to r_0$ & $count \to count_0$ at ①
let value of $p \to p_1$, $q \to q_1$, $r \to r_1$ & count → count1 at ②

2

If a,b,c hold at 1, then it should be true at 2

→ if ($p <= mid$ & $q <= j$ & $a[p] <= a[q]$)
then ($b[r] = a[p]$ $p++$, $r++$

(1) By (a) at 1 $b[i \ldots r_0-1]$ is sorted - 1
at 1 $b[i \ldots r_0-1] <= a[p]$ - 2
$b[r_0] = a[p]$ - 3
from 2 & 3
$b[i \ldots r_0]$ is sorted
At 3 $r = r_0 + 1$
⇒ $b[i \ldots r-1]$ is sorted

2 at 1 $b[i \ldots r_0-1]$ is permutation of $a[i \ldots p_0-1] \cup$
$a[mid+1 \ldots q_0-1]$.

at 2 $r = r_0+1$, $p = p_0+1$, $q = q_0$
so again $b[i \ldots r-1]$ is permutation of $a[i \ldots p-1]$
$\cup\, a[mid+1 \ldots q-1]$

3. at 2 $b[i \ldots r_0-1] <= a[p_0], a[q_0]$
$b[i \ldots r_0] <= a[q_0], a[p_0]$ (because
$b[r_0] = a[p_0]$
$<= a[q_0]$

* if $p_0 = mid$
$p = p_0+1 = mid+1$
& $b[i \ldots r_0] <= a[q_0]$
and $p = p_0+1$, $q = q_0$, $r = r_0+1$
So $b[i \ldots r-1] <= a[q]$

* if $p_0 < mid$.

Date
Page

$p = p_0 + 1 = mid + 1$

$b[i \ldots r_0] <= a[q_0], a[p_0]$

$a[p_0] <= a[p_0+1]$

$\Rightarrow a[i \ldots r_0] <= a[q_0], a[p_0+1]$

At (2) $p = p_0 + 1$, $r = r_0 + 1$, $q = q_0$

$\Rightarrow a[i \ldots r-1] \leq a[p], a[q]$

* $i <= p <= mid+1$, $mid+1 <= q <= j+1$, $r = p + q - mid - 1$

as $p \geq i$ & $q \geq mid+1$

$p_0 <= mid$, $q_0 <= j$ at (1)

at (2) ~~$q = p$~~ $p = p_0 + 1$ & $q = q_0$

$\Rightarrow p \leq mid+1$ $q <= j$

and $r_0 = p_0 + q_0 - mid - 1$

$\Rightarrow r_0 + 1 = p_0 + q_0 - mid$ — (x)

So (x) becomes $r = p + q - mid - 1$ — (✓) True

~~*(2) contains the n~~

(*) at (2) count contain no. of inversions (k, l)

$i <= l <= mid$ & $mid+1 \leq l \leq q_0 - 1$

at (2) $p = p_0 + 1$, $q = q_0$, $r = r_0 + 1$, $inv = inv_0$

As no change in value of q.

$\Rightarrow$ no change in 'count';

\# If $(p \leq mid$ & $q <= j$ & $a[p] > a[q])$

then $[\, b[r] = a[q]$, $count += (mid - p + 1)$, $q{+}{+}$, $r{+}{+}$

Proof of sorting algorithm is done in lectures.

So for proof of count of inversion at (1) & (2)

inv. of the form (l, k)

& $i <= l \leq mid$ & $mid+1 \leq k \leq q_0 - 1$

$a[p_0] > a[q_0]$

array $a[i -- mid]$ is sorted.

$a[p_0 --- mid] \geq a[p_0]$

$\Rightarrow a[p_0 --- mid] > a[q_0]$

* if $(p_0 > i)$

→ we have to prove $a[p_0 - 1] <= a[q_0]$

So by contradiction $a[p_0 - 1] > a[q_0]$

⇒ in previous iteration q would increase by 1

→ $q \to q_0 + 1$ at (2) which would result in $q_0 + 1$ at (1)

which is wrong.

So in array $a[i -- mid]$ only $a[p_0 -- mid]$ is greater than $a[q_0]$.

⇒ No. of inversion of form (l, q_0) where $i <= l <= mid$ will be number of elements in array $a[p_0 -- mid]$ which will be $(mid - p_0 + 1)$

⇒ no. of inv. $= count_0 + (mid - p_0 + 1)$ — (Y)

At (2) $q = q_0 + 1$, $r = r_0 + 1$, $inv = count_0 + (mid - p_0 + 1)$

So, from (Y)

$count$ is count in inv. of form (l, k) where $i <= l <= mid$ & $mid+1 <= k <= q-1$

Date

Page

if ($p > mid$ & $q <= j$)
then { $b[r] = a[q]$, $q{+}{+}$, $r{+}{+}$ }

→ at ① $b[i \ldots r_0 - 1]$ is sorted
$b[i \ldots r_0 - 1] <= a[q_0]$
New $b[r_0] = a[q_0]$
⇒ $b[i \ldots r_0 - 1] <= b[r_0]$ ⇒ $b[i \ldots r_0]$ is sorted

At ② $r = r_0 + 1$
$b[i \ldots r - 1]$ is sorted

→ at ① $b[i \ldots r_0 - 1]$ is permutation of $a[i \ldots p_0 - 1] \cup a[mid+1 \ldots q_0 - 1]$
$b[r_0] = a[q_0]$
⇒ $a[i \ldots r_0]$ is permutation of $a[i \ldots p_0 - 1]$ ~~ ~~
$\cup\ a[mid+1 \ldots q_0]$

At ② $r = r_0 + 1$, $p = p_0$, $q = q_0 + 1$
⇒ $b[i \ldots r - 1]$ is permutation of $a[i \ldots p - 1] \cup a[mid+1 \ldots q - 1]$

↪ $i <= p <= mid + 1$, $mid + 1 <= q \leq j + 1$, $r = p + q - mid - 1$
as $p \geq i$ & $q \geq mid + 1$
at ① $p_0 <= mid + 1$
$q_0 <= j$ at ①
at ② $q = q_0 + 1$ $p = p_0$
$q = q_0 + 1 <= j + 1$ & $p <= mid + 1$ at ②
at ① $r_0 = p_0 + q_0 - mid - 1$
$r_0 + 1 = p_0 + q_0 - mid$
at ③ $p = p_0$; $q = q_0 + 1$ $r = r_0 + 1$
→ $r = p + q - mid - 1$ ✓

→ Since $p > mid$. So no p in $a[i..mid]$ s.t. it is greater than $a[q]$ as no 'p' existed.

Hence for q_0 there is no l in $a[i..mid]$ s.t. pair(l, q_0) forms an inversion (no l s.t. $a[l] > a[q_0]$).

So no. increase in inversion

$count = count_0$

Also at (3) $p = p_0,\ q = q_0 + 1,\ r = r_0 + 1$ ~~final~~ $count = count_0$

\# if ($p <= mid$ & $q > j$)

then { $b[r] = a[p]$, $p{+}{+}$, $r{+}{+}$ }

THIS IS ANALOGOUS TO PREVIOUS IF STATEMENT

Now Correctness

at (2) $r = p + q - mid - 1$ & $p <= mid + 1$ $q <= j + 1$

$\Rightarrow r <= j + 1$ & $r > j$

$\Rightarrow r = j + 1$

if $r = j + 1$, then p & q must be max-possible

$\Rightarrow p = mid + 1$ & $q = j + 1$

→ $b[i..r-1]$ is sorted

→ $b[i..r-1]$ is permutation of $a[i..p-1] \cup a[mid+1..q-1]$

→ (i) $b[i..r-1] <= a[p], a[q]$ if $p <= mid$, $q <= j$

$<= a[q]$ if $p > mid$, $q <= j$

$<= a[p]$ if $p <= mid$, $q > j$

Date
Page

→ $i <= p <= mid+1$ $mid+1 \le q \le j+1$ $r = p+q-mid-1$

(i) $b[i..r-1]$ is sorted ⇒ $b[i..j]$ is sorted
⇒ $b[i..j]$ is permutation of $a[i..j]$.

count is inversion of form (l, k), where
$i \le l \le mid$ $mid+1 \le k \le j$

~~Func. 'Sort'~~ gives value of

Func. 'Merge' gives value of no. of inversions of form (l, k)
$i \le l \le mid$, $mid+1 \le k \le j$.
or all pairs $(a[l], a[k])$ for which $a[l] > a[k]$

Hence proved.

Now Correctness of func.ⁿ "Merge Sort"

Merge sort $(a[], i, j)$
if $(i >= j)$ {
—
--
}

Here in this func.ⁿ we count the total inversion
b/w index 'i' to 'mid'
and from index 'mid+1' to 'j'

And also inversion during sort and merge

Thus we get total number of inversions
from index 'i' to 'j'.