CS 60-454
Design and Analysis of Algorithms

ASSIGNMENT #2

FALL 2018

Due Date: November 6 **(before lecture)**

The following rules apply to all assignments handed out in this course.

- For every algorithm you present, you must include:
  1. a clear English description of the algorithm;
  2. the algorithm in a pseudo-code (possibly with examples showing how it works);
  3. a correctness proof of the algorithm;
  4. an analysis of the time complexity of the algorithm.

  Omitting any of the above, particularly items 3 or 4, would result in getting a 0 mark.
- Type your solutions if your hand-written is not legible.

1. Present an $\Theta(n \lg n)$–time algorithm that takes a list of elements drawn from a totally ordered set as input and reports the number of inversions in the list.

Solution: We modify Algorithm MERGESORT as follows.

Algorithm INVERSION(*L*, *lower*, *upper*, *icount*);
Input : *L*[*lower* .. *upper*];
Output : *L*[*lower* .. *upper*] sorted in ascending order;
$icount$: the number of inversions in L
begin
$icount$:= 0;
if (*lower* < *upper*) **then**
1. **INVERSION**(L, *lower*, $\left\lfloor \frac{(lower+upper)}{2} \right\rfloor$, *icnt*1);
2. **INVERSION**(L, $\left\lfloor \frac{(lower+upper)}{2} \right\rfloor$+1, *upper*, *icnt*2);
3. **Merge**($L\left[lower..\left\lfloor \frac{(lower+upper)}{2} \right\rfloor\right]$, $L\left[\left\lfloor \frac{(lower+upper)}{2} \right\rfloor + 1..upper\right]$, *icnt*3) into $L[lower..upper]$;
4. $icount := icnt1 + icnt2 + icnt3$

end.

Algorithm Merge$(A, B, Inversion)$;
Input: Two sorted lists *A*[1..*m*] and *B*[1..*n*];
Output: The number of Inversions in $A \oplus B$, where $\oplus$ is the concatenation operator, and the sorted $A \oplus B$.
begin
$index_A$:= 1; $index_B$:= 1; $Inversion := 0$;
while ($index_A \leq m$ and $index_B \leq n$) **do**

if ($A[index_A] \leq B[index_B]$)

then $C[index_C] := A[index_A]$; $index_A := index_A+1$;

else $C[index_C] := B[index_B]$; $index_B := index_B+1$;

$Inversion := Inversion + (m - index_A + 1)$;

$index_C := index_C + 1$

endwhile;

if ($index_A > m$)

then copy $B[index_B..n]$ into $C[index_C..m+n]$;

else copy $A[index_A..m]$ into $C[index_C..m+n]$

end.

Recall that (a_i, a_j) is an inversion if and only if $(i < j) \wedge (a_i > a_j)$.

Remark: For clarity and convinence, from here onwards, we shall use $L[1..n]$ instead of $L[lower..upper]$ to represent any sublist of L.

Lemma 1: Let (a_i, a_j) be an inversion in $L[1..n]$. Then $a_i, a_j \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right]$, or $a_i, a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$, or $a_i \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right] \wedge a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$.

Proof: Either $a_i \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right]$ or $a_i \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$.

Suppose $a_i \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right]$.

Then either $a_j \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right]$ or $a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$.

In the former case, $a_i, a_j \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right]$. In the latter case, $a_i \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right]$ and $a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$.

Suppose $a_i \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$.

Since $i < j$, we thus have $\left\lfloor\frac{1+n}{2}\right\rfloor + 1 < j \leq n$ which implies that $a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$. Hence, $a_i, a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$. $\square$

Owing to Lemma 1, the set of inversions in $L[1..n]$ can be partitioned into the following three disjoint subsets:

$I_1 = \left\{(a_i, a_j) \mid (i < j) \wedge (a_i > a_j) \wedge a_i, a_j \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right]\right\}$,

$I_2 = \left\{(a_i, a_j) \mid (i < j) \wedge (a_i > a_j) \wedge a_i, a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]\right\}$, and

$I_3 = \left\{(a_i, a_j) \mid (i < j) \wedge (a_i > a_j) \wedge a_i \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right] \wedge a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]\right\}$.

Lemma 2: Let $I_3' = \left\{(a_i, a_j) \mid (i < j) \wedge (a_i > a_j) \wedge a_i \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right] \wedge a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]\right\}$ be the set of inversions with $a_i \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right]$ and $a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$ after Step 2 and before Step 3 of Algorithm INVERSION is executed. Then $I_3' = I_3$.

Proof:

$\Leftarrow$) Let $(a, b) \in I_3$. Then $(\exists i, j)(a_i = a \wedge a_j = b)$ with $(i < j) \wedge (a_i > a_j)$ and $a_i \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right] \wedge a_j \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$.

It follows that $a \in L\left[1..\left\lfloor\frac{1+n}{2}\right\rfloor\right] \wedge b \in L\left[\left\lfloor\frac{1+n}{2}\right\rfloor + 1..n\right]$.

Let $a_k = a$ and $a_l = b$ after Step 2 and before Step 3 is executed.

Since a remains in $L\big[1..\lfloor\frac{1+n}{2}\rfloor\big]$ and b remains in $L\big[\lfloor\frac{1+n}{2}\rfloor+1..n\big]$, therefore, $a_k \in L\big[1..\lfloor\frac{1+n}{2}\rfloor\big]$ and $a_l \in L\big[\lfloor\frac{1+n}{2}\rfloor+1..n\big]$.

It follows that $(k < l) \wedge (a_k > a_l)$ which implies that $(a_k, a_l) \in I'_3$. Hence $(a, b) \in I'_3$.

$\Rightarrow)$ Similar to the above case. $\square$

Lemma 3: Algorithm Merge correctly counts the number of inversions in I'_3.

Proof: For each $a_l \in L\big[\lfloor\frac{1+n}{2}\rfloor+1..n\big]$, let $I'_{3_l} = \big\{(a_i, a_j) \in I'_3 \mid j = l\big\}$.

If $I'_{3_l} = \emptyset$, then there is no inversion in I'_3 involving a_l. It follows that $(\nexists i)(i < l \wedge a_i > a_l)$ and $a_i \in L\big[1..\lfloor\frac{1+n}{2}\rfloor\big]$. This implies that when $B[index_B] = a_l$, the **else** part of the **if** statement is never executed. Hence, the $Inversion$ counter correctly remains unchanged.

On the other hand, if $I'_{3_l} \neq \emptyset$, let a_k be the smallest element in $L\big[1..\lfloor\frac{1+n}{2}\rfloor\big]$ such that $a_k > a_l$.

As $L\big[1..\lfloor\frac{1+n}{2}\rfloor\big]$ is sorted in ascending order after Step 2, we must have $a_i \leq a_l, 1 \leq i < k$, and $a_i > a_l, k \leq i \leq \lfloor\frac{1+n}{2}\rfloor$. Therefore, there are exactly $\lfloor\frac{1+n}{2}\rfloor - k + 1$ inversions involving a_l in I'_{3_l}. As a result, when $B[index_B] = a_l$ and $A[index_A] = a_k$, as $a_k > a_l$, the **else** part of the **if** statement is executed which correctly increases the $Inversion$ counter by the amount of $m - index_A + 1 = \lfloor\frac{1+n}{2}\rfloor - k + 1$ and removes a_l from further consideration.

Hence, when the execution of Algorithm Merge terminates, $Inversion = \big|I'_3\big|$. $\square$

Theorem 4: Algorithm INVERSION correctly counts the number of inversions in the input list L.

Proof: (By induction on $|L|$)

Induction basis: When $|L| = 1$, The body of the **if** statement is never executed and the algorithm correctly returns 0 as the number of inversions in L.

Induction hypothesis: Suppose Algorithm INVERSION correctly counts the number of inversions for any input list L with $|L| < n$.

Induction step: Consider an input list L with $|L| = n$.

Since $\big|L\big[1..\lfloor\frac{1+n}{2}\rfloor\big]\big| < n$ and $\big|L\big[1..\lfloor\frac{1+n}{2}\rfloor\big]\big| < n$, by the induction hypothesis, Algorithm INVERSION correctly counts the number of inversions in both sublists. Therefore, after Step 2, $icnt1 = |I_1|$ and $icnt2 = |I_2|$ which are the number of inversions in $L\big[1..\lfloor\frac{1+n}{2}\rfloor\big]$ and $L\big[1..\lfloor\frac{1+n}{2}\rfloor\big]$, respectively.

By Lemma 3, $icnt3$ contain the number inversions in I'_3 which is also the number inversions in I_3 owing to Lemma 2. Therefore, $icnt3 = |I_3|$

As a result, the value of $icount$ returned by Algorithm INVERSION is $|I_1| + |I_2| + |I_3|$ which is the total number of inversions in the input list L. $\square$

Theorem 5: Algorithm INVERSION runs in $\Theta(n \lg n)$ time.

Proof: Since the modification made on Algorithm MERGESORT involves instructions that increase the time complexity by only a constant factor, the time complexity of Algorithm INVERSION is thus $\Theta(n \lg n)$. $\blacksquare$