Homework 2 (50 points) Due: September 20, 2024 11:59 pm
COMPSCI 733: Advanced Algorithms and Designs

Documentation: (5 points) Type your solutions using Latex (www.overleaf.com or https://www.latex-project.org/). Submit your solutions (pdf is enough) to Canvas.

Problem 1: (15 points) Assume we have a one dimensional array of real numbers with indices, $1, 2, \ldots, n$. The following pseudocodes for MERGE and MERGE-SORT are from CLRS textbook.

Algorithm 1 MERGE(A,p,q,r)

$n_1 = q - p + 1$
$n_2 = r - q$
let $L[1 \ldots n_1 + 1]$ and $R[1 \ldots n_2 + 1]$ be new arrays
for $i = 1$ to n_1 **do**
    $L[i] = A[p + i - 1]$
end for
for $j = 1$ to n_2 **do**
    $R[j] = A[q + j]$
end for
$L[n_1 + 1] = \infty$
$R[n_2 + 1] = \infty$
$i = 1$
$j = 1$
for $k = p$ to r **do**
    if $L[i] \leq R[j]$ **then**
        $A[k] = L[i]$
        $i = i + 1$
    else
        $A[k] = R[j]$
        $j = j + 1$
    end if
end for

Algorithm 2 MERGE-SORT(A,p,r)

if $p < r$ **then**
    $q = (p + r)/2$
    MERGE-SORT(A, p, q)
    MERGE-SORT$(A, q + 1, r)$
    MERGE(A, p, q, r)
end if

(a) Write pseudocodes with new parameters to modify the above MERGE and MERGE-SORT that divide the array into three equal parts, sort them, and do a three-way merge as follows.

Use a new parameter s and write $MERGE-3(A, p, q, r, s)$ and $MERGE-SORT-3(A, p, s)$, where A is an array and p, q, r, and s are indices into the array such that $p \leq q \leq r < s$. $MERGE$ function assumes that the subarrays $A[p, \ldots, q], A[q + 1, \ldots, r]$ and $A[r+1, \ldots, s]$ are in sorted order. Make sure to write the complete pseudo codes for the modified functions.

Algorithm 3 MERGE-3(A,p,q,r,s)

1: Your pseudocode goes here.

Algorithm 4 MERGE-SORT-3(A,p,s)

1: Your pseudocode goes here.

(b) Let $T(n)$ be the running time of MERGE-SORT-3 on an array of size n. Write a recurrence relation (i.e., $T(n) =?$) for your algorithm.

In Problem 2 and 3, you need to complete the details of a proof for the correctness of Merge Sort. You need to refer to the above Algorithm 2, $MERGE\_SORT(A, P, r)$, and Algorithm 1, $MERGE(A, p, q, r)$, given in the CLRS textbook. We will establish correctness of Merge-sort in two parts:

1. Assuming correctness of the Merge procedure, prove the correctness of MergeSort.
2. Prove the correctness of the Merge procedure.

Problem 2: (15 points)

Prove: Assuming that the procedure for Merge is correct, a call to $Merge\_Sort(A, p, r)$, $p \leq r$, returns the elements in $A[p..r]$ rearranged in sorted order, and does not alter any entry outside of the subarray $A[p..r]$.

Proof. : By strong induction on $m = r-p+1$ (i.e., by induction on the size of the sub-array $A[p..r]$).

Base case: Need to show correctness for the smallest input size. This occurs when $p = r$, i.e., when $m = 1$. Since the sub-array $A[p..r]$) has only one element, it is trivially sorted.

Induction step. Assume correctness for all sizes $1 \leq k < m$, and establish correctness when $k = m$. **Fill in the details of this induction step.** □

Problem 3: (15 points) Correctness of Merge.

Merge procedure copies subarray $A[p..q]$ into $L[1..n1]$ and $A[q+1..n2]$ into $R[1..n2]$, with $L[n1+1]$ and $R[n2+1]$ set to ∞ (a value larger than any of the elements in $A[p..r]$).

Correctness of Merge is established through the correctness of the following loop invariant for the for loop in Line 14 of the above Algorithm 1:

Loop invariant:

At the start of each iteration of the for loop in line 14, $A[p..k-1]$ contains the $k-p$ smallest elements in $L[1..n1+1]$ and $R[1..n2+1]$ in sorted order. Further, $L[i]$ and $R[j]$ are the smallest elements in their arrays that have not been copied back into A. The elements in array A outside of subarray $A[p..r]$ are unchanged.

Complete the following steps:

1. Show Initialization holds.This establishes the base case by proving that the loop invariant holds just before the start of the first iteration.

2. Show Maintenance holds.Assuming that the loop invariant holds at the start of a given iteration this establishes that it continues to hold at the start of the next iteration.

3. Show Termination holds.States what the loop invariant establishes about the computation at the time when the loop is exited.