CSC 225 FALL 2022
ALGORITHMS AND DATA STRUCTURES I
ASSIGNMENT 4 - SOLUTIONS
UNIVERSITY OF VICTORIA

1. [4 marks] The following is my code and analysis. They will all be different. There should be no instance of an array element being compared to an array element.

Algorithm stackSort(A,n):
    Input: Array A with n values
    Output: Array A, sorted

    Stacks S_1, S_2
    for $k \leftarrow 0$ **to** $n-1$ **do**
        while !S_1.isEmpty() **and** $A[k] < S_1$.top() **do**
            S_2.push(S_1.pop())
        end
        S_1.push($A[k]$)
        while !S_2.isEmpty() **do**
            S_1.push(S_2.pop())
        end
    end
    for $k \leftarrow 0$ **to** $n-1$ **do**
        $A[k] \leftarrow S_1$.pop()
    end
end

2

2

In the first for loop, in the worst-case, we will move k elements from S_1 to S_2, push 1 element onto S_1, and then move k elements from S_2 to S_1, for each k. That is, $2k+1$ operations for each k from 0 to $n-1$. That is a total of

$$\begin{aligned} T(n) &= (2(0)+1)+(2(1)+1)+(2(2)+1)+\cdots+(2(n-1)+1) \\ &= n + 2\big(1+2+\cdots+(n-1)\big) \\ &= \cancel{n+n}(n-1) \\ &= n^2 \end{aligned}$$

operations.

The second loop does n assignments. Thus, this is an $O(n^2)$ sorting algorithm that takes $O(n)$ extra space for the two stacks.

2. [4 marks] For each of the iterative algorithms I will only indicate when swaps occur. I will mark the elements that will move and then move them.

<u>Selection Sort:</u>

$$A = \left[\underbrace{5}, 7, \underbrace{0}, 3, 4, 2, 6, 1\right]$$
$$A = \left[0, \underbrace{7}, 5, 3, 4, 2, 6, \underbrace{1}\right]$$
$$A = \left[0, 1, \underbrace{5}, 3, 4, \underbrace{2}, 6, 7\right]$$
$$A = [0, 1, 2, 3, 4, 5, 6, 7]$$

<u>Bubble Sort:</u>

$$A = \left[5, \underbrace{7}, \underbrace{0}, 3, 4, 2, 6, 1\right]$$
$$A = \left[5, 0, \underbrace{7}, \underbrace{3}, 4, 2, 6, 1\right]$$
$$A = \left[5, 0, 3, \underbrace{7}, \underbrace{4}, 2, 6, 1\right]$$
$$A = \left[5, 0, 3, 4, \underbrace{7}, \underbrace{2}, 6, 1\right]$$
$$A = \left[5, 0, 3, 4, 2, \underbrace{7}, \underbrace{6}, 1\right]$$
$$A = \left[5, 0, 3, 4, 2, 6, \underbrace{7}, \underbrace{1}\right]$$
$$A = \left[\underbrace{5}, \underbrace{0}, 3, 4, 2, 6, 1, 7\right]$$
$$A = \left[0, \underbrace{5}, \underbrace{3}, 4, 2, 6, 1, 7\right]$$
$$A = \left[0, 3, \underbrace{5}, \underbrace{4}, 2, 6, 1, 7\right]$$
$$A = \left[0, 3, 4, \underbrace{5}, \underbrace{2}, 6, 1, 7\right]$$
$$A = \left[0, 3, 4, 2, 5, \underbrace{6}, \underbrace{1}, 7\right]$$
$$A = \left[0, 3, \underbrace{4}, \underbrace{2}, 5, 1, 6, 7\right]$$
$$A = \left[0, 3, 2, 4, \underbrace{5}, \underbrace{1}, 6, 7\right]$$
$$A = \left[0, \underbrace{3}, \underbrace{2}, 4, 1, 5, 6, 7\right]$$
$$A = \left[0, 2, 3, \underbrace{4}, \underbrace{1}, 5, 6, 7\right]$$
$$A = \left[0, 2, \underbrace{3}, \underbrace{1}, 4, 5, 6, 7\right]$$
$$A = \left[0, \underbrace{2}, \underbrace{1}, 3, 4, 5, 6, 7\right]$$
$$A = [0, 1, 2, 3, 4, 5, 6, 7]$$

Insertion Sort:

$A = [5, 7, 0, 3, 4, 2, 6, 1]$

$A = [0, 5, 7, 3, 4, 2, 6, 1]$

$A = [0, 3, 5, 7, 4, 2, 6, 1]$

$A = [0, 3, 4, 5, 7, 2, 6, 1]$

$A = [0, 2, 3, 4, 5, 7, 6, 1]$

$A = [0, 2, 3, 4, 5, 6, 7, 1]$

$A = [0, 1, 2, 3, 4, 5, 6, 7]$

Mergesort:

$A = [5, 7, 0, 3, 4, 2, 6, 1]$

$A_1 = [5, 7, 0, 3]$ $\quad$ $A_2 = [4, 2, 6, 1]$

$A_1 = [5, 7]$ $\quad$ $A_2 = [0, 3]$ $\quad$ $A_1 = [4, 2]$ $\quad$ $A_2 = [6, 1]$

$A_1 = [5]$ $\quad$ $A_2 = [7]$ $\quad$ $A_1 = [0]$ $\quad$ $A_2 = [3]$ $\quad$ $A_1 = [4]$ $\quad$ $A_2 = [2]$ $\quad$ $A_1 = [6]$ $\quad$ $A_2 = [1]$

$A_1 = [5, 7]$ $\quad$ $A_2 = [0, 3]$ $\quad$ $A_1 = [2, 4]$ $\quad$ $A_2 = [1, 6]$

$A_1 = [0, 3, 5, 7]$ $\quad$ $A_2 = [1, 2, 4, 6]$

$A = [0, 1, 2, 3, 4, 5, 6, 7]$

3. [4 marks] Let A be a sorted array of size n, and consider the element at position $\lfloor n/2 \rfloor$. If n is even, then there are $n/2$ elements $< A[\lfloor n/2 \rfloor]$ and $n/2 - 1$ elements $> A[\lfloor n/2 \rfloor]$. If n is odd, then there are $\lfloor n/2 \rfloor$ elements $< A[\lfloor n/2 \rfloor]$ and $\lfloor n/2 \rfloor$ elements $> A[\lfloor n/2 \rfloor]$. Either way, $T(n) \leq 2T(n/2) + n$ for values of $n \geq 1$. We have seen that this recurrence equation is $O(n \log n)$.

2

To run in $\Theta(n^2)$ time, the element at index $\lfloor n/2 \rfloor$ would have to be either the largest or smallest element in the array, for every recursive call. In that case either L or G has $n-1$ elements in it and the other would have 0 elements. An example of such an array would be $A = [6,4,2,1,3,5]$. I will illustrate with a recursion tree trace of this example below, where I have bolded the pivot at each step.

2

$A = [6, 4, 2, \mathbf{1}, 3, 5]$

- $L = [\ ]$
- $G = [6, 4, \mathbf{2}, 3, 5]$
  - $L = [\ ]$
  - $G = [6, 4, \mathbf{3}, 5]$
    - $L = [\ ]$
    - $G = [6, \mathbf{4}, 5]$
      - $L = [\ ]$
      - $G = [6, \mathbf{5}]$
        - $L = [\ ]$
        - $G = [6]$

4. [4 marks] We will use a loop invariants proof to show that insertion sort is correct. Our goal is to prove that,

S: Insertion sort sorts an n-element array, A, in increasing order.

To do this we want to prove that S_k is true at the end of iteration k, for each $k = 1, \ldots, n-1$, where,

1/2 S_k: The first $k+1$ elements of array A, ($A[0]$ to $A[k]$) are sorted in increasing order.

1 for base case + hypothesis

So, S_0 states that the first element ($A[0]$) is sorted before the first iteration of the loop, which is trivially true. Assume that S_{k-1} is true before the k^{th} iteration, and consider S_k. During the k^{th} iteration, val is assigned $A[k]$ and j is assigned $k-1$. From there one of two cases arise:

1/2

1. val $\geq A[j]$, that is, $A[k] \geq A[k-1]$. In this case, we do not enter the inner loop and $A[j+1]$ is assigned val, i.e. $A[k]$ stays where it is and $A[0]$ to $A[k]$ are sorted. Thus, S_k is true.
2. val $< A[j]$, that is, $A[k] < A[k-1]$ and we enter the inner loop. The invariant on the inner loop at the end of iteration j, for each $j = k-1, \ldots, 0$, is, 1/2

2

T_j: Array elements $A[j], \ldots, A[k] \geq$ val.

Clearly, T_k is true as val is assigned to $A[k]$ before entering the inner loop and so trivially, $A[k] \geq$ val. If we assume T_{j+1} is true for some $j+1 \leq k-1$, then consider T_j. Since T_{j+1} is true, we know that $A[j+1], \ldots, A[k] \geq$ val. At the beginning of the current iteration, if $j \geq 0$ and

1

$A[j] >$ val, then we assign $A[j]$ to $A[j+1]$ and it is true that $A[j], \ldots, A[k] \geq$ val and so T_j is true. Note, if either $j < 0$ or $A[j] \leq$ val we do not enter the inner loop.

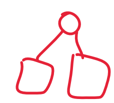
This implies, that once we end the inner loop, whatever the current j may be, $A[j+1], \ldots, A[k] \geq$ val, and $A[0], \ldots., A[j] <$ val, (note, if $j < 0$ then nothing is less than val.) Also, every element from $j+1$ to $k-1$ has been assigned one position to the right in A. Then, we assign val to $A[j+1]$ and A[0] to A[k] are sorted in increasing order. That is, S_k is true.

The outer loop (and thus the algorithm) terminates when k=n and thus we know by our loop invariant that S_{n-1} is true, that is, The first n elements of array A, ($A[0]$ to $A[n-1]$) are sorted in increasing order. This implies that A is sorted and insertion sort is correct. 1/2

5. [4 marks] Let T is a proper binary tree with n internal nodes, $I(T)$ the sum of the depths of all the internal nodes of T and $E(T)$ the sum of the depths of all the external nodes. We will show that $E(T) = I(T) + 2n$ using strong induction on the number of internal nodes, n. For the purposes of strong induction, we will consider two base cases for our proof, when $n = 0$ and $n = 1$.

1 Base Case: When $n = 0$, then T is a single external node with depth 0 and no internal nodes. Thus, $E(T) = 0$, $I(T) = 0$ and $I(T) + 2(0) = 0 + 0 = 0$, so $E(T) = I(T) + 2n$.

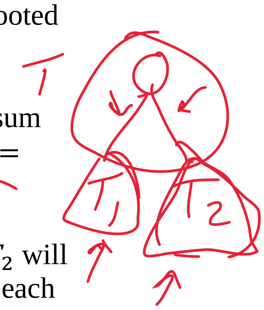
When $n = 1$, then T consists of a single internal node at the root with two children that are external nodes. The internal node is at depth 0 and the two external nodes are each at depth 1. Thus, $E(T) = 1 + 1 = 2$, $I(T) = 0$ and $I(T) + 2(1) = 2$, so $E(T) = I(T) + 2n$.



$n=k$ $2 = 0 + 2(1)$

1 Induction Hypothesis: Let $n \leq k$ and assume that $E(T) = I(T) + 2n$ for tree T with n internal nodes.

Induction Step: Now consider tree T, with $n = k + 1$ internal nodes. Remove from T the root node and the two edges that connect to its children. The result is two proper binary trees, T_1 and T_2, rooted at the children of tree T's root, respectively.



2 Let k_1 be the number of internal nodes in T_1 and k_2 be the number of internal nodes in T_2. The sum $k_1 + k_2 = k$, since T has $k+1$ internal nodes, and so both $k_1, k_2 \leq k$. This implies that $E(T_1) = I(T_1) + 2k_1$ and $E(T_2) = I(T_2) + 2k_2$, by the induction hypothesis.

Now, when we reconstruct T (reconnect the root of T back to T_1 and T_2), every node in T_1 and T_2 will get one edge deeper. That is, the total external and internal path lengths will increase by one for each external and internal node, respectively. Note, T_1 has $(k_1 + 1)$ external nodes and T_2 has $(k_2 + 1)$ external nodes (proven in class). So,

$$\begin{aligned}
E(T) &= E(T_1) + E(T_2) + (k_1 + 1) + (k_2 + 1) \\
&= (I(T_1) + 2k_1) + (I(T_2) + 2k_2) + (k_1 + 1) + (k_2 + 1) \\
&= (I(T_1) + I(T_2) + k_1 + k_2) + 2k_1 + 2k_2 + 2 \\
&= I(T) + 2(k_1 + k_2 + 1) \\
&= I(T) + 2(k + 1) \\
&= I(T) + 2n
\end{aligned}$$