

# Homework 2

ECE345 - Group 16

October 13th, 2023

Total pages: TBD

Team Member	Student Number
Ardavan Alaei Fard	1007934620
Rowan Honeywell	1007972945
Isaac Muscat	1007897135

## Contents

<b>Question 1</b>	<b>2</b>
(a) . . . . .	2
(b) . . . . .	2
(c) . . . . .	2
<b>Question 2</b>	<b>3</b>
(a) . . . . .	3
(b) . . . . .	3
<b>Question 3</b>	<b>4</b>
(a) . . . . .	4
(b) . . . . .	5
(c) . . . . .	5
<b>Question 4</b>	<b>6</b>
(a) . . . . .	6
(b) . . . . .	6
(c) . . . . .	7
(d) . . . . .	7
(e) . . . . .	8

## Question 1

(a)

(b)

(c)

## Question 2

(a)

(b)

## Question 3

(a)

Assume  $S_n$  represents sequence  $S$  with size of  $n$  elements

---

**Algorithm 1** Calculate intersection of two sorted sequences

---

**Require:**  $n, A_n, B_n \mid A[i] \leq A[j] \vee B[i] \leq B[j]$  for  $0 \leq i \leq j \leq n - 1$

**Ensure:**  $I = A_n \cap B_n$

```
 $I \leftarrow$  empty list
 $a \leftarrow 0$ 
 $b \leftarrow 0$ 
while  $a < n$  and  $b < n$  do
  if  $A[a] = B[b]$  then
     $I.insert(A[a])$ 
     $a \leftarrow a + 1$ 
     $b \leftarrow b + 1$ 
  else if  $A[a] < B[b]$  then
     $a \leftarrow a + 1$ 
  else if  $A[a] > B[b]$  then
     $b \leftarrow b + 1$ 
  end if
end while
if  $a = n$  then
  for  $b$  to  $n - 1$  do
    if  $B[b] > B[b - 1]$  and  $A[n - 1] = B[b]$  then
       $I.insert(B[b])$ 
    end if
  end for
else if  $b = n$  then
  for  $a$  to  $n - 1$  do
    if  $A[a] > A[a - 1]$  and  $A[a] = B[n - 1]$  then
       $I.insert(A[a])$ 
    end if
  end for
end if
```

---

(b)

(c)

## Question 4

Note: Leftist heap will be abbreviated as LH.

(a)

Let  $s$  represent the smallest complete sub-tree of an LH  $L$  starting from the root. Since the rank of  $L$  will be the length of the shortest path from the root to the leaf, the height of  $s$  will have a height of the rank of the root of  $L$  (otherwise the  $s$  would not be complete). If  $m$  is the number of nodes in  $s$ , the height of  $s$  will be  $\mathcal{O}(\lg m)$  which will be the same as the rank of the root. If  $n$  is the number of nodes in  $L$ , then since  $s$  is a sub-tree of  $L$ ,  $n \geq m \implies \lg n \geq \lg m \implies$  the rank of the root of an LH is  $\mathcal{O}(\log n)$ . QED

(b)

From (a), we know that the rank of the root of an LH is  $\mathcal{O}(\log n)$  which is the same as the length of the rightmost path. We also know that to merge two sorted sequences using **MERGE** (CLRS, 4th, page 38), it takes  $\Theta(n)$ . If the size of two leftist heaps  $l_1$  and  $l_2$  have sizes  $n_1$  and  $n_2$ , then to merge the rightmost paths of  $l_1$  and  $l_2$ , the **MERGE** procedure will have to iterate over  $\mathcal{O}(\log n_1) + \mathcal{O}(\log n_2) = \mathcal{O}(\log n)$  elements. Therefore, to merge  $l_1$  and  $l_2$ , it takes  $\mathcal{O}(\log n)$  time. To show that the order invariant is maintained, suppose that an LH  $l_1$  with  $rank = 0$  is being added to the right child of the root of another LH  $l_2$  with its right child removed in the LH merge procedure where merging two LHs splits both of them into sub-trees with their root's right child removed. Since the key of the root of  $l_1$  is larger and all other nodes of  $l_1$  are larger than its root by the definition of an LH, all other nodes in  $l_1$  will be larger than the root of  $l_2$ . QED

(c)

Since merging the rightmost paths of two trees  $t_1$  and  $t_2$  involves recursively removing the right child from the produced sub-trees, merging the rightmost paths of two trees can be modelled as merging the right path of  $m$  sub-trees without a right child at their roots. After splitting  $t_1$  and  $t_2$  into  $m$  sub-trees, only the right child of the root of each sub-tree will be changed. Take one of the subtrees  $m_1$ . If a new tree of arbitrary shape and size is set as the right child of  $m_1$ , then the left child subtree of  $m_1$  would not have been changed. Since the rank of a node is defined as  $1 + \min(\text{rank}(\text{left}(x)), \text{rank}(\text{right}(x)))$  and the left subtree of  $m_1$  was not changed, the rank does not change as well. In contrast, the rank of the right child of  $m_1$  will be of arbitrary size. QED

(d)

From (b) we know that the process of merging the rightmost path of two LHs takes  $\mathcal{O}(\log n)$  time and that it keeps the order invariant. Since the rank update step is executed after their rightmost paths are merged, we need to show that the rank update step takes  $\mathcal{O}(\log n)$  time and that it maintains the balance invariant.

Suppose we are left with a LH  $l$  that is the result of merging the rightmost path of two LHs. We will prove the balance invariant is maintained after the rank update step using induction where  $P(n+1)$  represents the parent of the node. Basis: the rightmost leaf subtree of  $l$  has no children so it has a rank of 0 which is the lowest. Induction Hypothesis: If the children of the root of the subtree are LHs, then swapping the children of the root of the subtree if the rank of the left child is greater than the rank of the right child will turn the subtree into a LH. Inductive Step: From (c) we know that merging the rightmost paths will only change



the rank of nodes on the rightmost path. If we apply the update step, the right child will have a smaller rank. Since the right child is an LH, the left child is an LH, and the length of the rightmost path of the subtree root is the smallest, the subtree will be an LH. In

From (a), we also know that the length of the rightmost path of the merged LH will be  $\mathcal{O}(\log n_1) + \mathcal{O}(\log n_2) = \mathcal{O}(\log n)$ . If the rightmost path of the merged tree is traversed from the bottom rightmost leafnode, then  $\log n$  nodes will be traversed with their children being optionally swapped. Since swapping the child of each parent is just swapping two pointers which takes  $\Theta(1)$  time, the rank update step takes  $\mathcal{O}(\log n)$  time.

(e)

To implement **DeleteMin** and Insert, we can utilize the **Merge** procedure that runs in  $\mathcal{O}(\log n)$  time. The subtree starting at each node of an LH is also an LH or else the order and balance invariant would not be maintained which is why we can use the merge procedure here.

DELETEMIN( $H$ )

```
1   $l = \text{LEFTIST-HEAP}(H.\text{root}.\text{left})$ 
2   $r = \text{LEFTIST-HEAP}(H.\text{root}.\text{right})$ 
3   $H = \text{MERGE}(l, r)$ 
```

INSERT( $H, i$ )

```
1   $H = \text{MERGE}(H, \text{LEFTIST-HEAP}(H.i))$ 
```

## References

- [1] CLRS, a, b, c, d