CSCI 470 Homework 3

Please write your solutions in the LaTeX. You may use online compiler such as Overleaf or any other compiler you are comfortable with to write your solutions in the LaTeX.

Due date: Sunday Oct 22, 11:59 PM EDT.

Please submit a PDF (preferably written in LaTeX) or a scanned copy of your handwritten solutions to Homework 03 on Canvas.

You can use the LaTeX submission template I have shared along with the homework. There are two .tex files ("macros.tex", and "main.tex"). You can upload the zipped folder directly to Overleaf or create a blank project on Overleaf and upload macros.tex and main.tex files, and edit main.tex to write your solutions. "macros.tex" is mostly for macros (predefined commands).

Handwritten solutions will also be accepted. Points will be deducted if handwritten solutions are not legible.

Sorting in Linear Time [15 points]

Problem 3-1. (10 points) Obtain asymptotically tight bounds on $\lg(n!)$ without using Stirling's approximation. Instead, evaluate the summation $\sum_{k=1}^{n} \lg k$ using techniques from Section A.2 (CLRS 3rd edition pg 1149 - 1154).

Problem 3-2. (5 points) Using Figure 8.2 (pg 195) as a model, illustrate the operation of COUNTING-SORT on the array $A = \langle 6, 0, 2, 0, 1, 3, 4, 6, 1, 2, 2 \rangle$.

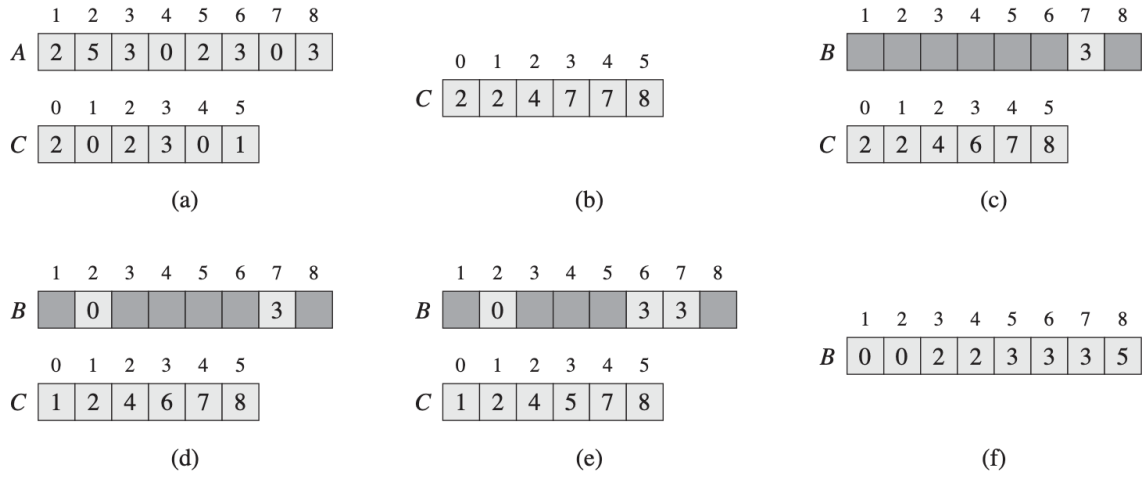


Figure 8.2 The operation of COUNTING-SORT on an input array $A[1..8]$, where each element of A is a nonnegative integer no larger than $k = 5$. **(a)** The array A and the auxiliary array C after line 5. **(b)** The array C after line 8. **(c)–(e)** The output array B and the auxiliary array C after one, two, and three iterations of the loop in lines 10–12, respectively. Only the lightly shaded elements of array B have been filled in. **(f)** The final sorted output array B.

Stack and Queues [30 points]

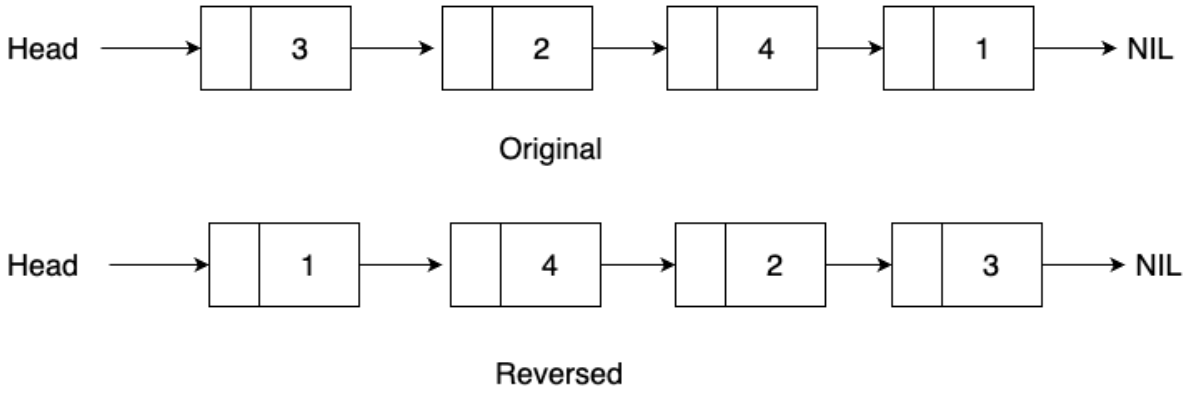
Problem 3-3. (10 points) Rewrite ENQUEUE and DEQUEUE to detect underflow and overflow of a queue.

Problem 3-4. (20 points) Whereas a stack allows insertion and deletion of elements at only one end, and a queue allows insertion at one end and deletion at the other end, a ***deque*** (double-ended queue) allows insertion and deletion at both ends. Write four $O(1)$-time procedures to insert elements into and delete elements from both ends of a deque implemented by an array.

1. (5 points) HEAD-ENQUEUE(Q, x)
2. (5 points) TAIL-ENQUEUE(Q, x)
3. (5 points) HEAD-DEQUEUE(Q, x)
4. (5 points) TAIL-DEQUEUE(Q, x)

Linked Lists and Binary Trees [15 points]

Problem 3-5. (10 points) Give a $\Theta(n)$-time **nonrecursive** procedure that reverses a singly linked list of n elements. The procedure should use no more than constant storage beyond that needed for the list itself.



Problem 3-6. (5 points) Write an $O(n)$-time **recurisve** procedure that, given an n-node binary tree, prints out the keys of each node in the tree. *Please note that the order of the printed keys does not matter here*.

Hash Tables [20 points]

Problem 3-7. (5 points) Suppose that a dynamic set S is represented by a direct-address table T of length m. Describe a procedure that finds the maximum element of S. What is the worst-case performance of your procedure?

Problem 3-8. (10 points) Demonstrate what happens when we insert the keys 5, 28, 19, 15, 20, 33, 12, 17, 10 into a hash table with collisions resolved by chaining. Let the table have 9 slots and let the hash function be $h(k) = k \mod 9$.

Problem 3-9. (5 points) Consider a hash table of size $m = 1000$ and a corresponding hash function $h(k) = \lfloor m(kA \mod 1) \rfloor$ for $A = (\sqrt{5} - 1)/2$. Compute the locations to which the keys 61, 62, 63, 64, and 65 are mapped.

Binary Search Trees [40 points]

Problem 3-10. (5 points) What is the difference between the binary-search-tree property and the min-heap property (see page 153)? (5 points) Can the min-heap property be used to print out the keys of an n-node tree in sorted order in $O(n)$ time? Show how, or explain why not.

Problem 3-11. (10 points) Give **recursive** algorithms that perform preorder and postorder tree walks in $\Theta(n)$ time on a tree of n nodes.

Problem 3-12. (5 points) Write **recursive** versions of TREE-MINIMUM and TREE-MAXIMUM

Problem 3-13. (5 points) Write the TREE-PREDECESSOR procedure.

Problem 3-14. (10 points) Give a **recursive** version of the TREE-INSERT procedure.

Extra Credit [35 points]

Problem 3-15. (10 points) Write an $O(n)$-time **nonrecursive** procedure that, given an n-node binary tree, prints out the key of each node in the tree. Use a stack as an auxiliary data structure.

Problem 3-16. (10 points) Suppose that we are storing a set of n keys into a hash table of size m. Show that if the keys are drawn from a universe U with $|U| > nm$, then U has a subset of size n consisting of keys that all hash to the same slot, so that the worst-case searching time for hashing with chaining is $\Theta(n)$.

Problem 3-17. (15 points) Suppose that instead of each node x keeping the attribute $x.p$, pointing to x's parent, it keeps $x.succ$, pointing to x's successor. Give pseudocode for DELETE on a binary search tree T using this representation. This procedure should operate in time $O(h)$, where h is the height of the tree T. (*Hint:* You may wish to implement a subroutine that returns the parent of a node.)