# CSCI2100C 2019-20: Assignment 3*

# This assignment is due at 11:59:59pm, 12th April 2020.

- **Q1. [14 marks]** AVL-Tree.
  - **(i). [10 marks]** Given an empty AVL-Tree, show the AVL-Tree after inserting 4, 8, 9, 2, 3, 7, 6 in order and each insertion operation step by step. (Refer to CSCI2100C-Lecture13 Pages 20-21)
  - **(ii). [4 marks]** Given an AVL-Tree as shown in Figure 1, show how to delete the node 2 in this AVL-Tree step by step. (Refer to CSCI2100C-Lecture13 Pages 24-27)

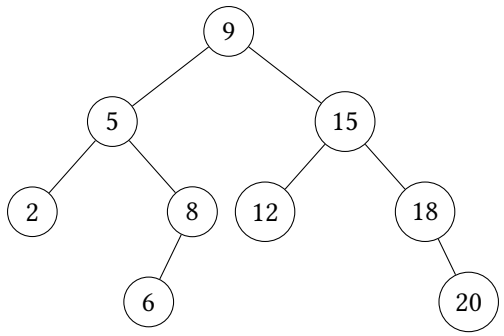


Figure 1. An AVL-Tree for Q1

- **Q2. [14 marks]** Treap.
  - **(i). [10 marks]** Draw the treap after inserting nodes 2, 3, 5, 7, 11 in order, assuming that their respective priorities are 6, 5, 7, 3, 8. Show each insertion operation step by step. (Refer to CSCI2100C-Lecture14 Pages 14-15)
  - **(ii). [4 marks]** Given a treap as shown in Figure 2, where the value above the line is the key and the one below the line is the priority in a node, show how to delete node 7 (the node whose key is 7) in this treap step by step. (Refer to CSCI2100C-Lecture14 Pages 17-18)

---

*Departmental Guideline for Plagiarism (Department of Systems Engineering and Engineering Management): If a student is found plagiarizing, his/her case will be reported to the Department Examination Panel. If the case is proven after deliberation, the student will automatically fail the course in which he/she committed plagiarism. The definition of plagiarism includes copying of the whole or parts of written assignments, programming exercises, reports, quiz papers, mid-term examinations and final examinations. The penalty will apply to both the one who copies the work and the one whose work is being copied, unless the latter can prove his/her work has been copied unwittingly. Furthermore, inclusion of others' works or results without citation in assignments and reports is also regarded as plagiarism with similar penalty to the offender. A student caught plagiarizing during tests or examinations will be reported to the Faculty office and appropriate disciplinary authorities for further action, in addition to failing the course.

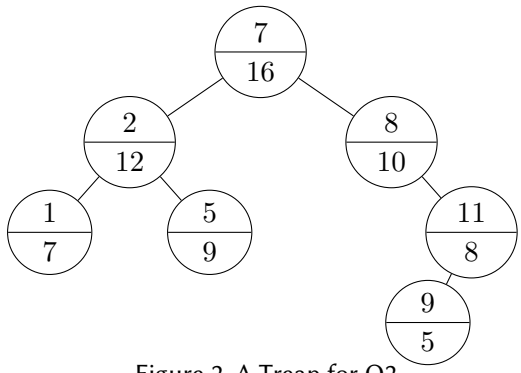

Figure 2. A Treap for Q2

- **Q3. [12 marks]** Sort an array $A[1 \cdots 8] = [4, 6, 3, 8, 5, 2, 7, 9]$ in ascending order by heap sort.
  - **(i). [6 marks]** Show the contents of $A$ in the heap adjust process to make it a max-heap step by step. (Refer to CSCI2100C-Lecture17 Pages 6-8)
  - **(ii). [6 marks]** Using the max-heap of part (i), show the contents of $A$ in the sorting process of swaping elements in the array step by step. (Refer to CSCI2100C-Lecture17 Page 9)
- **Q4. [22 marks]** Answer the following questions about merge sort and quicksort.
  - **(i). [8 marks]** Given an unsorted sequence $[4, 1, 6, 3, 2, 9, 5, 7]$, please fill in the diagram of Figure 3 step by step to show how to sort the sequence with merge sort. (Some contents of the diagram have been given in Figure 3.) (Refer to CSCI2100C-Lecture15-16 Pages 16-25)

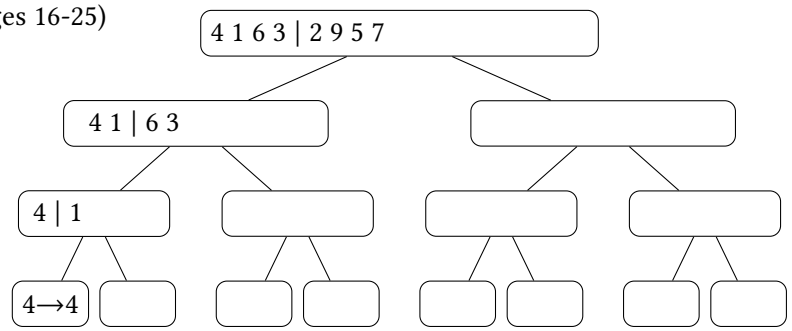

Figure 3. A Diagram of Mergesort for Q4

  - **(ii). [6 marks]** We use a **temparray** (Refer to CSCI2100C-Lecture15-16 Pages 26-28) to store the merge result. Given an array $A[0 \cdots 7] = [1, 3, 6, 8, 2, 5, 7, 9]$, during the invocation of **merge**($A$, 0, 3, 7), show the contents of **temparray** and indicate the values of **left** and **secondleft** pointing the original array $A$ step by step.
  - **(iii). [8 marks]** Let $A[0 \cdots 7] = [6, 3, 5, 2, 7, 4, 1, 8]$. Assume that we call **quicksort**($A$, 0, 7) (Refer to CSCI2100C-Lecture15-16 Pages 34-36) and the **pivot** (i.e., the position of the randomly chosen element in $A$) chosen randomly is 2. Show what happens in the contents of $A$ and indicate the values of **nextsmallpos** during the invocation of **partition**($A$, 0, 7, 2) step by step.

- **Q5. [10 marks]** Show how to use the counting sort to sort an array $A[0 \cdots 9] = [6, 1, 2, 1, 3, 4, 6, 1, 3, 2]$ with $U = 6$ (i.e., the keys are integers within the range $[1, 6]$) step by step. (Refer to CSCI2100C-Lecture17 Pages 27-32)
- **Q6. [6 marks]** Based on the idea of counting sort, describe an algorithm that, given $n$ integers in the range $[1, k]$, preprocesses its input and then answers any query about how many of the $n$ integers fall into a range $[a, b]$ ($a$ and $b$ are integers in the range $[1, k]$) in $O(1)$ time. Your algorithm should use $O(n + k)$ preprocessing time. Justify your answer.
- **Q7. [14 marks]** Radix Sort. (Refer to CSCI2100C-Lecture18 Page 5 and Page 8)
  - **(i). [8 marks]** Show the sorting results after sorting on each digit with radix sort for an array $A[0 \cdots 6] = [4650, 7391, 5479, 3152, 2465, 3918, 9830]$.
  - **(ii). [6 marks]** Show how to sort $n$ integers in the range $[1, n^3 - 1]$ in $O(n)$ time. Justify your answer.
- **Q8. [8 marks]** You are given $n$ distinct integers in an array **arr**. For finding the $k$-th smallest element in this array, there is a random algorithm shown as below. Based on this algorithm, answer the following questions.

```
Algorithm kthSmallest(arr, left, right, k)
1:  if left == right
2:     return arr[left]
3:  pivot = random(left, right)
4:  pivotnewpos = partition(arr, left, right, pivot)
5:  while pivotnewpos ∉ [left+1/3·(right-left+1), left + 2/3·(right-left+1)]
6:     pivot = random(left, right)
7:     pivotnewpos = partition(arr, left, right, pivot)
8:  if pivotnewpos-left == k-1
9:     return arr[pivotnewpos]
10: elseif pivotnewpos-left > k-1
11:    return kthSmallest(arr, left, pivotnewpos-1, k)
12: else
13:    return kthSmallest(arr, pivotnewpos+1, right, (k-1)-(pivotnewpos-left))
```

  - **(i). [4 marks]** During an invocation of **kthSmallest(arr, left, right, k)**, we assume that the size of the subarray **arr[left...right]**, i.e., **right-left+1**, is an integer $n$ that is multiple of 3. For this invocation, calculate the expected number of the basic operation **random(left, right)** executed from Lines 1-7. Besides, what is the expected running cost for Lines 1-7 in terms of Big-Oh? Justify your answer. (Refer to CSCI2100C-Lecture15-16 Pages 34-36 for functions of **random**, **partition** and definitions of the **pivot** and **pivotnewpos**.
  - **(ii). [4 marks]** Let $T(n)$ be the expected number of comparisons in the above algorithm for finding the $k$-th smallest element in an array of size $n$. Derive the recurrence of $T(n)$ and analyze the asymptoic complexity of $T(n)$ in terms of Big-Oh. Justify your answer. (Hint. Consider the largest size of the subarray **arr[left, pivotnewpos-1]** (or **arr[pivotnewpos+1, right]**) when we do the recursion in Line 11 (or Line 13))