

National University of Singapore

School of Computing

CS2040S - Data Structures and Algorithms

Final Assessment

(Semester 1 AY2024/25)

Time Allowed: 2 hours

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains THREE (3) sections.
It comprises EIGHTEEN (18) printed pages, including this page.
3. This is an **Open Book Assessment**.
Only non-programmable calculator is allowed in this assessment.
4. Answer **ALL** questions within the **boxed space** of the answer sheet (page 13-18).
For Section A, shade the option in page 13 of the answer sheet (use 2B pencil).
There are a few starred (*) boxes: free 1 mark if left blank but 0 for wrong answer (no partial).
The answer sheet is at page 13-18.
You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
Read all the questions first! Some (subtask) questions might be easier than they appear.
6. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.
You can use **standard, non-modified** classic algorithm in your answer by just mentioning its name, e.g. run BFS on graph G , Dijkstra's on graph G' , etc.
7. The total marks is 100. All the best :)

A MCQs ($20 \times 2 = 40$ marks)

Select the **best unique** answer for each question. Each correct answer worth 2 marks.

- What is the *tightest worst-case* time complexity of the following Java function in terms of n ?
 Notice the keyword ‘tightest’, i.e., if the best answer is x but you choose an option that is worse than x (which is still true in Big O notation, but not the tightest), you will be marked as wrong.
 Similarly if you choose an option that is better than x (impossible), you will be marked as wrong.

```
static int foo(ArrayList<Integer> something) {
    int to_report = 0, n = something.size();
    Collections.sort(something);
    for (int z : something)
        if (z % 2 != 0)
            ++to_report;
    return to_report;
}
```

- a). $O(n)$
 - b). $O(n \log n)$
 - c). $O(n^2)$
 - d). $O(n^2 \log n)$
 - e). $O(n^3 \log n)$
- Refer to the same code as in Question 1 above. If **ArrayList** z contains the following Integers $\{78, 46, 9, 57, 78, 40, 52, 50, 99, 15\}$, then **foo(z)** will return:
 - a). 5
 - b). 6
 - c). 7
 - d). 8
 - e). None of the above
 - Which problem on an **ArrayList** A of $n \geq 2$ Integers does **not** become **easier** (i.e., admits faster solution) after we sort A (versus if A remains unsorted)? Note: No additional data structure.
 - a). Finding duplicates in A
 - b). Finding two Integers in A that are consecutive (i and $i + 1$)
 - c). Finding the second smallest Integer in A
 - d). Finding if A contains at least one odd Integer
 - e). Finding the index of value v in A , if it exists

4. You are working on a UNIX terminal. Your working directory is currently ‘/home/s/stevenha’ which can be accessed by using ‘pwd’ (print working directory) command. You navigate directories using ‘cd `directory`’ (changes the working directory to `directory`, note that there is no character ‘/’ in `directory`) and ‘cd ..’ (changes the working directory to the one above the current working directory, i.e., goes back one directory). You are given N (a lot) directory navigation commands, i.e., ‘cd `directory`’ or ‘cd ..’ commands. At the end of these N directory navigation commands, you are asked to execute ‘pwd’. Which ADT that you will use to support these $N + 1$ commands efficiently?
- a). Stack ADT
 - b). Queue ADT
 - c). Priority Queue ADT
 - d). Table ADT
 - e). Graph ADT
5. Which underlying data structure *cannot* be used to achieve $O(\log n)$ or better for both enqueue and dequeue operations of a (Min) Priority Queue ADT (of n distinct 64-bit signed Integers)?
- a). Bucket Queue as discussed in Section B.2
 - b). Binary (Max) Heap
 - c). Binary (Min) Heap
 - d). Java `PriorityQueue` class
 - e). Balanced Binary Search Tree
6. We use a Hash Table of size $m = 17$ to store $n = 7$ Integers. We use hash function $h(v) = v \% m$. We resolve collision using *Linear Probing*. The Hash Table is initially empty. Which sequence of insertion of these n Integers will result in at least one Integer using **at least** 2 linear probing steps when inserted into the Hash Table?
- a). {19, 5, 40, 96, 33, 83, 4}
 - b). {1, 2, 3, 4, 5, 6, 7}
 - c). {33, 39, 12, 98, 31, 15, 33}
 - d). {88, 72, 22, 25, 46, 97, 63}
 - e). None of the above

7. Starting from a new instance of UFDS with $N = 13$ disjoint sets, we call `unionSet(7, 2)`, `unionSet(8, 1)`, `unionSet(0, 9)`, `unionSet(3, 2)`, `unionSet(7, 2)` (this is a repeat), `unionSet(2, 5)`, and `unionSet(4, 1)`. Note that both path-compression and union-by-rank heuristics are used. Which statement is **incorrect** about the current state of the UFDS?
- It currently has 7 disjoint sets
 - There is one disjoint set with 4 members
 - There are 4 disjoint sets with just 1 member
 - Item 7 and 0 belong to different disjoint sets
 - There is one vertex with rank 2.
8. What is the highest possible actual height of an AVL Tree with $N = 88$ Integers?
 Height of the tree is defined as the number of edges from root to the deepest leaf.
- 5
 - 6
 - 7
 - 8
 - 13
9. Which of the following *unweighted* graph is *not implicit*? An implicit graph can have their edges derived on-the-fly when running a Graph Algorithm without the need of a specific Graph DS.
- 2D Square Grid Graph of size $r \times c$; an edge connects 2 cells if they share a border
 - Knight Jump graph of 8×8 chessboard
 - The Forest of Trees of Union-Find Disjoint Sets
 - Complete Binary Tree of n vertices of Binary Max Heap
 - Complete Bipartite Graph of n/m vertices on the left/right set, respectively;
 Left set contains vertices $[1..n]$ and Right set contains vertices $[n + 1..n + m]$
10. Which Graph DS below is the best to store a directed weighted graph with n vertices and m edges (up to $\min(10\,000, n \times (n - 1)/2)$ edges). The graph algorithm needs to frequently query the current weight of a directed edge $u \rightarrow v$ and to update its weight as fast as possible. ($1 \leq n \leq 1\,000\,000$).
- Adjacency Matrix (AM)
 - Adjacency List (AL)
 - Edge List (EL)
 - Implicit Graph (the graph does not have to be explicitly stored)
 - Hash Table with key (u, v) and value the current weight w

11. On what kind of input graph can Bellman-Ford algorithm **possibly fail** to produce the correct shortest path values from a given source vertex s ?
- An undirected **non-negative** weighted Tree
 - An undirected non-negative weighted graph
 - A Directed *Acyclic* Graph (DAG)
 - A directed non-negative weighted graph
 - None of the above
12. On what kind of input graph can (the Original form of) Dijkstra's algorithm **possibly fail** to produce the correct shortest path values from a given source vertex s ?
- An undirected non-negative weighted Tree
 - A directed (may have cycles) weighted graph where all edges are positive
 - A directed *acyclic* weighted graph where all edges are non-negative
 - A directed *acyclic* weighted graph where there can be some edges that are negative
 - None of the above
13. How many valid topological sorts can you find in the DAG shown in Figure 1?

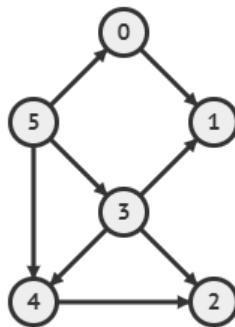


Figure 1: The Graph for Q13+14+15

- 6
 - 7
 - 8
 - 9
 - None of the above
14. Which statement about Figure 1 is incorrect?
- The graph is unweighted
 - The graph is acyclic

- c). The shortest path from vertex 5 to 2 has 2 edges
d). The longest path from vertex 5 to 2 has 4 edges
e). None of the above
15. Which of the following is **not** a valid DFS traversal of the graph shown in Figure 1?
Note that the DFS visitation order depends on how we order the neighbors.
DFS is started from the source vertex $s = 5$.
- a). 5, 0, 1, 3, 2, 4
b). 5, 0, 1, 4, 2, 3
c). 5, 3, 1, 2, 4, 0
d). 5, 4, 2, 3, 0, 1
e). 5, 4, 2, 3, 1, 0
16. The input graph is an **undirected connected (non-negative) weighted graph** that have lots of trivial cycles (bidirectional edges (u, v)) but it has no cycle involving three or more vertices. Which graph algorithm is the most suitable (correct and the fastest) to find the shortest path between a distinguished source vertex s of that input graph and another vertex t ?
- a). Depth First Search
b). Dijkstra's algorithm (either version)
c). Topological sorting algorithm
d). Bellman-Ford algorithm
e). Actually, all algorithms above are equally good to solve the given problem
17. Which algorithm cannot be used to solve the MST problem in $O(E \log V)$ time?
- a). Prim's algorithm
b). Kruskal's algorithm
c). Boruvka's algorithm
d). Jarnik's algorithm
e). All algorithms above can solve the MST problem in similar time complexity
18. How many of the five graph algorithms below runs in $O(V + E)$?
- i. DFS
ii. BFS
iii. Dijkstra's (original version)
iv. Kahn's
v. Cycle detection with DFS

- a). 1
- b). 2
- c). 3
- d). 4
- e). 5

19. How many of the five statements below are **True**?

- i. An undirected graph cannot have a cycle
 - ii. There exists at least one Directed Acyclic Graph (DAG) that has *exactly one, i.e., unique* topological sort
 - iii. Dijkstra's algorithm (either version) can be used to solve weighted SSSP problem even if the underlying graph has a negative weight cycle reachable from the source vertex s
 - iv. A Bipartite Graph is also a Tree
 - v. We can test if vertex u and v of an undirected graph G are connected via a path (or not) by running **the original version of** Dijkstra's algorithm with u as the source vertex and check if $dist[v] \neq \infty$
- a). 1
 - b). 2
 - c). 3
 - d). 4
 - e). 5

20. How many of the five statements below are **False**?

- i. The time complexity of running DFS algorithm on a weighted tree is $O(V)$
 - ii. BFS algorithm cannot be implemented faster than $O((V + E) \log V)$
 - iii. Every connected undirected graph G has a spanning tree
 - iv. Every undirected graph with V vertices and $V - 1$ edges is always a Tree
 - v. The SSSP spanning tree found by Dijkstra's algorithm and MST found by Prim's algorithm rooted at the same source vertex s on the same weighted graph can never be identical
- a). 1
 - b). 2
 - c). 3
 - d). 4
 - e). 5

B Questions with Short Answers (20 marks)

B.1 Space-Time Trade-off ($3 + 3 + 4 = 10$ marks)

In most Data Structures and Algorithms courses, we focus on time complexity analysis. However, space complexity analysis can also be useful at certain scenarios.

In this question, you are asked to come up with *three* different algorithms for the same problem: Write a function `static boolean can(int N, ArrayList<Integer> A, int T)` to decide if the given array A (not necessarily sorted) that contains N integers has two distinct indices i and j such that $A[i] + A[j] = T$.

B.1.1 Time: $O(N^2)$, (Additional) Space: $O(1)$

Design an algorithm that is allowed to run in $O(N^2)$ time but uses not more than $O(1)$ *additional* space (other than $O(N)$ of array A itself), i.e., you can use a few **constant** variables.

B.1.2 Time: (Expected) $O(N)$, (Additional) Space: $O(N)$

Design an algorithm that must run in (expected) $O(N)$ time but can use up to $O(N)$ *additional* space (on top of $O(N)$ of array A itself), i.e., you can process array A **not** in-place.

B.1.3 Time: (Expected) $O(N \log N)$, (Additional) Space: (Expected) $O(\log N)$

Design an algorithm that must run in (expected) $O(N \log N)$ time but only use up to (expected) $O(\log N)$ *additional* space (on top of $O(N)$ of array A itself).

B.2 Bucket Queue ($5 + 5 = 10$ marks)

In class, we learn about Binary Max Heap data structure (<https://visualgo.net/en/heap>) that can be used to implement Priority Queue (PQ) ADT (of any keys that are Comparable) such that `Insert(v)` and `ExtractMax()` can both be done in $O(\log N)$. Can we do even better? We actually can, if the priorities of the PQ ADT are **Integers** $\in [0..C]$ and C is “**reasonably small**”. Introducing: Bucket Queue.

A Bucket Queue basic form is `ArrayList<Integer> A` of size $C + 1$, initially set to all zeroes. The value of `A.get(p)` is the frequency of Integer key p in the PQ ADT. So, we can implement `Insert(v)` by simply incrementing the frequency at index v by $+1$. We can implement `ExtractMax()` as a sequential search from C down to 0 to find the highest index p whereby the frequency at that index is non-zero. If index p is found, we decrement the frequency at that index p by -1 (or report -1 if the maximum is not found).

B.2.1 Write Java Code for `Insert(v)` and `ExtractMax()`

Given the pseudocode above, write the Java Code (**note that you are already given the pseudocode, so answering in pseudocode does not make sense for this question**) for these two crucial PQ ADT operations. Analyze the worst-case time complexity of the implementation in terms of N and/or C .

B.2.2 Write Java Code for `Update(oldV, newV)` and `Delete(oldV)`

Show that Bucket Queue, as long as the terms and conditions are satisfied, can also be used to do these additional two PQ ADT operations that is much more complicated to be implemented using Binary (Max/Min) Heap data structure. Note: Simply do nothing if `oldV` is not found. **If there are multiple copies of `oldV`, only update/delete one copy.** Analyze the worst-case time complexity of the implementation in terms of N and/or C .

C Applications (40 marks)

C.1 Generate Minimum-Sized AVL Tree (10 marks)

The minimum-sized AVL Tree is a height-balanced AVL Tree with height h that has the minimum possible number of vertices N_h . Now, write `static ArrayList<Integer> generate(int h)` function that returns a sequence of N_h distinct Integers $\in [1..N_h]$ such that inserting this sequence into an initially empty AVL Tree, produces a minimum-sized AVL Tree of height h **without triggering any rotation**. Here are a few examples:

`generate(0)` should just return `{1}`.

`generate(1)` can return either `{2, 1}` or `{1, 2}`, as inserting these two integers in that sequence into an empty AVL Tree produces Figure 2.



Figure 2: A Minimum-Sized AVL Tree with $h = 1$, using sequence $\{2, 1\}$

`generate(2)` can return either $\{3, 2, 4, 1\}$, $\{3, 4, 2, 1\}$, $\{2, 1, 3, 4\}$, or $\{2, 3, 1, 4\}$ as inserting these four integers in that sequence into an empty AVL Tree produces Figure 3. Notice that the sequence $\{3, 2, 1, 4\}$ is wrong as after 1 is inserted, there will be a rotation happening.

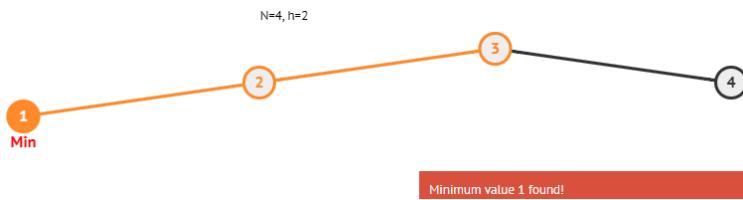


Figure 3: A Minimum-Sized AVL Tree with $h = 2$, using sequence $\{3, 2, 4, 1\}$

`generate(3)` can possibly return $\{5, 3, 7, 2, 6, 4, 1\}$ (there are other possible sequences), as inserting these seven integers into an empty AVL Tree produces Figure 4.

C.1.1 Draw Minimum-Sized AVL Tree with $h = 5$ (5 marks)

Do as asked.

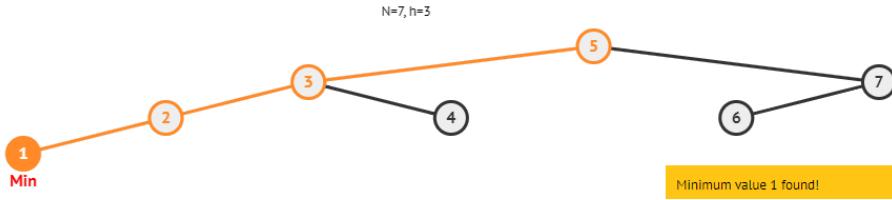


Figure 4: A Minimum-Sized AVL Tree with $h = 3$, using sequence $\{5, 3, 7, 2, 6, 4, 1\}$

C.1.2 Write generate(h) Function (5* marks)

Do as asked.

For this question, you can ignore time complexity analysis and focus on correctness, i.e., you can assume that $0 \leq h \leq 20$ (generate short Minimum-Sized AVL Tree).

C.2 Median Filter (15 marks)

An N -Integers ArrayList A of noisy data can often be smoothed out using a median filter. To do this we take a window of length K (K is guaranteed to be odd and $3 \leq K \leq N$) Integers and slide it over A , at each location we obtain the median Integer within the window.

The first window covers Integers from index 0 to index $K - 1$ and the smoothed data will contain $N - K + 1$ integers.

For example, given $A = \{70, 3, 5, 8, 6, 35, 7\}$ with $N = 7$ and $K = 3$, applying such median filter will produce the following result $A' = \{5, 5, 6, 8, 7\}$ with $7 - 3 + 1 = 5$ Integers because:

- The median of $\{70, 3, 5\} = 5$,
- The median of $\{3, 5, 8\} = 5$,
- The median of $\{5, 8, 6\} = 6$,
- The median of $\{8, 6, 35\} = 8$, and finally
- The median of $\{6, 35, 7\} = 7$.

C.2.1 Check Your Understanding (2 marks)

Given $B = \{92, 58, 15, 62, 31, 44, 95, 67, 66, 37\}$ with $N = 10$ and $K = 5$, smooth it out using median filter. What is the resulting smoothed array B' ?

C.2.2 Subtask 1: $K \leq N \leq 100\,000, K \in \{3, 5, 7\}$ (3 marks)

Design any algorithm that works when $K \in \{3, 5, 7\}$. You do not have to worry about K can be any odd Integer as big as N (which can be up to 100 000). Analyze the worst-case time complexity of your solution in terms of N and/or K .

Your algorithm must be fast enough (using not more than 10^8 operations in the worst-case).

You are free to skip this Subtask 1 if you are confident that you can do the full solution.

Note: Subtask 1 has different solution than Subtask 2.

C.2.3 Subtask 2: $K \leq N \leq 1000, 3 \leq K \leq N$ (5 marks)

Design any algorithm that works when K can be any odd Integer as big as N but N is not that big. Analyze the worst-case time complexity of your solution in terms of N and/or K .

Your algorithm must be fast enough (using not more than 10^8 operations in the worst-case).

You are free to skip this Subtask 2 if you are confident that you can do the full solution.

Note: Subtask 2 has different solution than Subtask 1.

C.2.4 Solve The Full Problem (5* marks)

Design any algorithm for this median filter problem that works for $K \leq N \leq 100\,000, 3 \leq K \leq N$ constraints. You are free to use any data structure that you see fit to do this task. Analyze the worst-case time complexity of your solution in terms of N and/or K . Your algorithm must be fast enough (using not more than 10^8 operations in the worst-case).

C.3 ASCII Art (15 marks)

You are given a 2D grid G of size $R \times C$ that describes a black-and-white ASCII art ($1 \leq R, C \leq 500$). Each cell in the grid is either a '#' (a black pixel) or a '.' (a white pixel). You say that two black '#' cells in the grid are connected if they share a border, i.e., one '#' cell is on the North/East/South/West of the other '#' cell (notice that diagonals are excluded).

Then, there are Q ($1 \leq Q \leq 100\,000$) `draw(i, j)` operations where $1 \leq i \leq R$ and $1 \leq j \leq C$ (notice 1-based indexing). It is guaranteed that $G[i][j] = \text{'.}'$ before this `draw(i, j)` operation and will turn to $G[i][j] = \text{'#}'$ afterwards. Your task is to print Q Integers: For each of this `draw(i, j)`, output the current size of the Connected Component that contains cell (i, j) .

For example, if you are given the following 5×6 grid:

```
.....#
.###..
.#####
..##..
.....
```

If you draw $Q = 2$ times, with the first one is `draw(5, 5)`, your ASCII art becomes:

```
.....#
.###..
.#####
..##..
....#. <-- output 1 as (5, 5) is not connected with any other black pixel
```

And if your second one is `draw(4, 5)`, your ASCII art becomes (note that this is the starting state for Subsection C.3.1):

```
.....# <-- not counted as it is not connected with (4, 5)
.###..
.#####
..###. <-- output 11 as (4, 5) is connected with 3+4+2+1 = 10 others
....#.
```

C.3.1 Check Your Understanding (2 marks)

Suppose we have 2 additional draw operations after the last state above: `draw(2, 5)` and then `draw(1, 5)`. What are the two additional Integers that you have to output?

C.3.2 Subtask 1: $R = 1$ (i.e., the grid is $1 \times C$) and $Q = 1$ (3 marks)

Design an $O(C)$ solution for this subtask.

If you are confident that you can solve the full version (Section C.3.4), you can skip this Subtask 1.

C.3.3 Subtask 2: $1 \leq R, C \leq 500$ (2D Grid) and $Q = 1$ (5 marks)

Design an $O(R \times C)$ solution for this subtask.

If you are confident that you can solve the full version (Section C.3.4), you can skip this Subtask 2.

C.3.4 Solve the Full Problem (5* marks)

Use all the insights above and design a correct algorithm to fully solve this problem: 2D Grid of size $1 \leq R, C \leq 500$ and lots of pixels drawn $1 \leq Q \leq 100\,000$. Analyze the worst-case time complexity of your proposed algorithm.

Your algorithm must be fast enough (using not more than 10^8 operations in the worst-case).

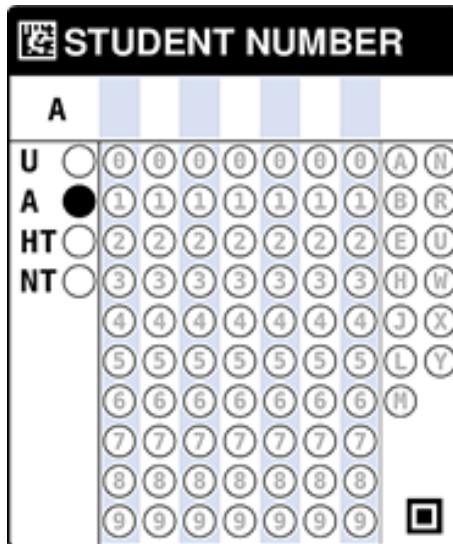
Note that if your full solution is correct, it should also solve Subtask 1+2 earlier.

Notice the grading scheme for this full problem, thus do attempt Subtask 1+2 first.

The Answer Sheet

Write your Student Number in the box below using **(2B) pencil**.

Do NOT write your name.



Write your MCQ answers in the special MCQ answer box below for automatic grading.

We do not manually check your answer.

Shade your answer properly (use (2B) pencil, fully enclose the circle; select just one circle).

No	1	2	3	4	5	6	7	8	9	10
A	<input type="radio"/>									
B	<input type="radio"/>									
C	<input type="radio"/>									
D	<input type="radio"/>									
E	<input type="radio"/>									

No	11	12	13	14	15	16	17	18	19	20
A	<input type="radio"/>									
B	<input type="radio"/>									
C	<input type="radio"/>									
D	<input type="radio"/>									
E	<input type="radio"/>									

Box B.1.1. $O(N^2)$ time, $O(1)$ additional space

Box B.1.2. (Expected) $O(N)$ time, $O(N)$ additional space

Box B.1.3. (Expected) $O(N \log N)$ time, (Expected) $O(\log N)$ additional space

Box B.2.1 and B.2.2. Complete the following Java code (do not forget to analyze time complexity):

```
import java.util.*;
public class B2 {
    public static void main(String[] args) {
        BucketQueue PQ = new BucketQueue(10); // Integer keys in [0..10]
        PQ.Insert(7); PQ.Insert(5); PQ.Insert(7);
        PQ.Insert(3); PQ.Insert(1); PQ.Insert(2);
        PQ.Insert(9);
        PQ.Update(7, 9); // now two copies of 9, only one 7
        PQ.Update(6, 9); // no effect as 6 does not exist
        PQ.Delete(1); // no 1 now
        PQ.Delete(6); // no effect as 6 does not exist
        while (true) {
            int maxV = PQ.ExtractMax();
            if (maxV == -1) break;
            System.out.print(maxV + ", ");
        } // prints out 9,9,7,5,3,2,
    }
}
```

```
class BucketQueue {  
    private static ArrayList<Integer> A;  
    private static int C;  
    public BucketQueue(int _C) {  
        C = _C;  
        A = new ArrayList<>(Collections.nCopies(C+1, 0));  
    }  
    public void Insert(int v) { // complete this and analyze time complexity  
  
    }  
    public int ExtractMax() { // complete this and analyze time complexity  
  
    }  
    public void Update(int oldV, int newV) { // complete this and analyze  
  
    }  
    public void Delete(int oldV) { // complete this and analyze  
  
    }  
}
```

Box C.1.1. Draw Minimum-Sized AVL Tree with $h = 5$

Box C.1.2*. (1 if blank, 0 if wrong) Complete the following Java code:

```
import java.util.*;
public class C1 {
    static ArrayList<Integer> generate(int h) { // complete this
}
public static void main(String[] args) {
    for (Integer i : generate(3))
        System.out.print(i + ",");
} // prints out 5,3,7,2,6,4,1, (or other valid sequence)
}
```

Box C.2.1 Check Your Understanding (write down the content of B')

Box C.2.2. $1 \leq N \leq 100\,000, K \in \{3, 5, 7\}$

Box C.2.3. $1 \leq N \leq 1000, 3 \leq K \leq N$

Box C.2.4*. (1 if blank, 0 if wrong) Solve the Full Problem and analyze time complexity

Box C.3.1 Check Your Understanding (write two additional Integers)

Box C.3.2. $R = 1$ (i.e., the grid is $1 \times C$) and $Q = 1$

Box C.3.3. $1 \leq R, C \leq 500$ (2D Grid) and $Q = 1$

Box C.3.4* (1 if blank, 0 if wrong) Solve the Full Problem and analyze time complexity

– END OF PAPER; All the Best –