CUHK(SZ)-CSC3100 Final Exam Sample

2nd Semester, 20xx-20yy

Note:

a. No notes or calculators are allowed in the exam.

c. Answer all questions within 120 minutes in an **answer book**.

1. (20 points) Choose **ONE** solution that best suits each question. (2 points each)

   (1) Which of the following sorting algorithms in its typical implementation gives the best performance when applied to an array that is sorted or almost sorted (maximum 1 or 2 elements are misplaced)?

   A. Insertion Sort

   B. Heap Sort

   C. Merge Sort

   D. Radix Sort

   (2) Which of the following are components of Hashing?

   A. Key

   B. Hash Function

   C. Hash Table

   D. All of the above

   (3) Which algorithm typically has the most space requirement on the same input?

   A. Selection Sort

   B. Insertion Sort

   C. Merge Sort

   D. Quick Sort

   (4) Which of the following data structures allows insertion and deletion from both ends?

   A. Stack

   B. Dequeue

   C. Queue

   D. Strings

   (5) Which searching technique takes $O(1)$ time complexity for searching the data?

   A. Linear search on an arbitrary array

   B. Binary search on a sorted array

   C. Search on a binary search tree

   D. Hashing on a hash table without collisions

   (6) ...

2. (10 points) Answer the following questions with either **true** or **false** (2 points each).

   (1) One can implement a stack based on a linked list so that each individual push/pop operation is in time $O(1)$.

   (2) One can implement a stack based on a linear array so that each individual push/pop operation is in time $O(1)$.

   (3) The core data structure used in Depth First Search is a queue.

   (4) One can reverse the order of the elements in a linked list in time $O(n)$.

   (5) It is possible to append two singly linked lists in time $O(1)$.

3. (15 points) Concisely answer the following short questions (3 points each).

   (1) List the names of three simple and stable sorting algorithms, and show their time complexities.

   (2) What is the primary conceptual difference between a stack and a queue?

   (3) Which data structures are typically used in Breadth First Search (BFS) and Depth First Search (DFS) algorithms, respectively?

   (4) What is the merge sort? How does it work?

   (5) What are the drawbacks of using a linked list to implement a linear queue?

4. (5 points) For the directed graph shown in Figure 1, draw both the adjacency matrix and adjacency list representations of this graph.

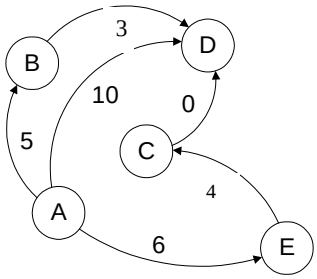

Figure 1: A Directed Graph.

5. (10 points) Suppose the class **java.util.LinkedList** is implemented by a doubly linked list, maintaining a reference to the first and last node in the list, along with its size.

```
public class LinkedList<Item> {
   private Node first;   // the first node in the linked list
   private Node last;    // the last node in the linked list
   private int N;        // number of items in the linked list
   private class Node {
      private Item item;         // the item
      private Node next, prev;   // next and previous nodes
   }
   ...
}
```

What are the best estimates of the worst-case running times of the following operations in big-O notation? (choose among $O(1), O(\log N), O(N), O(N \log N), O(N^2)$)

(1) **addFirst(item)**: Add the item to the beginning of the list.

(2) **get(i)**: Return the item a position i of the list.

(3) **set (i, item)**: Replace position i of the list with item.

(4) **removeLast()**: Delete and return the item at the end of the list.

(5) **contains(item)**: Is the item in the list?

6. (10 points) Consider a binary tree shown in Figure 2. For each of the **preorder**, **inorder** and **postorder** traversals, give the order in which the nodes are visited.

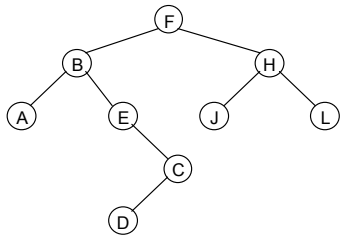

Figure 2: A Binary Tree.

7. (10 points) The height of a binary search tree (BST) depends on the order in which the keys are inserted into a tree if no balancing operation is performed. Given an initially empty BST, in what order will you insert the keys **A, B, C, D, E, F, G** so that the height of the BST is minimal. Note: the keys are in **alphabetic** order, i.e., $A < B < C < D < E < F < G$.

8. (10 points) A connected component of a graph is a set of nodes where each node can reach every other node in the component along the given edges, and which is connected to no additional nodes. For example, the graph in Figure 3 has three connected components.

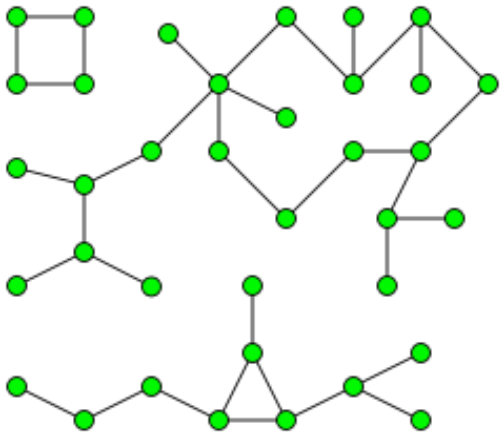

Figure 3: An Undirected Graph with Three Connected Components.

Explain, in words, how to use Kruskal's algorithm to compute the number of connected components in an undirected graph.

9. (10 points) The algorithm (pseudo code) in Figure 4 sorts an array (given as parameter *seq*) of n numbers. Estimate the time complexity of the algorithm as a function of input size n. Briefly show your calculation. Note: The function parameter *seq* is passed by reference, and $2n/3$ will be rounded up to the nearest integer in execution.

```
def triplesort(seq):
        if n <= 1: return
        if n == 2:
                replace seq by [min(seq),max(seq)]
                return
        triplesort(first 2n/3 positions in seq)
        triplesort(last 2n/3 positions in seq)
        triplesort(first 2n/3 positions in seq)
```

Figure 4: A Triple Sort Algorithm.