ĐẠI HỌC TÔN ĐỨC THẮNG TON DUC THANG UNIVERSITY	FACULTY OF INFORMATION TECHNOLOGY FINAL EXAMINATION SEMESTER 1 ACADEMIC YEAR: 2023 - 2024

SUBJECT: DATA STRUCTURES AND ALGORITHMS

SUBJECT CODE: 504008

REVISION FOR THE FINAL EXAMINATION

I. SORTING

Given an array of integers

[89, 40, 46, 55, 54, 5, 50, 73, 23, 47]

1) Present steps to sort the array in **ascending/descending** order using **Bubble Sort**.
2) Present steps to sort the array in **ascending/descending** order using **Selection Sort**.
3) Present steps to sort the array in **ascending/descending** order using **Insertion Sort**.
4) Present steps to sort the array in **ascending/descending** order using **Merge Sort**.
5) Implement, in Java, the method in Task (1).
6) Implement, in Java, the method in Task (2).
7) Implement, in Java, the method in Task (3).
8) Implement a class **MyComparator** that helps to sort an array of integers so that even numbers are all before odd numbers, even numbers are sorted ascendingly, and odd numbers are sorted descendingly.

```
class MyComparator implements Comparator<Integer> {}
```

II. RECURSION

1) Implement a recursive function to print down the binary form of a positive integer.
2) Implement a recursive function to reverse a positive integer.
3) Implement a recursive function to count the number of occurrences of a character in a string.
4) Implement a recursive function to convert a positive integer in decimal into hexadecimal. The function returns a string.
5) Implement a recursive function to print down items in an array of integers at index 1, 2, 4, 8, …, 2^k, …
6) Implement a recursive function to print down even digits of a positive integer. For example, a = 123456, print order: 6 4 2.

III. BINARY SEARCH TREE & AVL TREE

a) Binary Search Tree

Given a list of keys [40, 46, 55, 54, 5, 50, 73, 23, 47, 89]

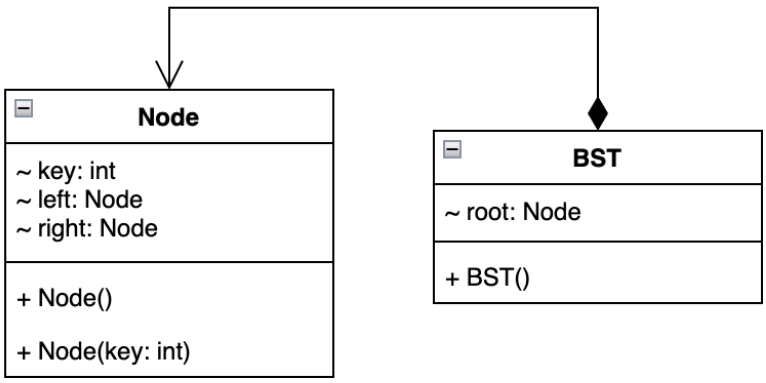
1) Present steps to build up a Binary Search Tree.
2) Delete leaf nodes.
3) Delete nodes with one child.
4) Delete nodes with two children using successors/predecessors.

b) AVL Tree

Given a list of keys [40, 46, 55, 54, 5, 50, 73, 23, 47, 89]

1) Present steps to build up a AVL Tree.
2) Delete leaf nodes.
3) Delete nodes with one child.
4) Delete nodes with two children using successors/predecessors.

c) Implementation



Given the class diagram above. Students implement, in Java, recursive functions below to perform the designated tasks.

1) Count the number of leaves
2) Compute the size of a subtree
3) Compute the sum of keys whose values in the range [a, b], given a and b.
4) Count the number of nodes that have one child only.

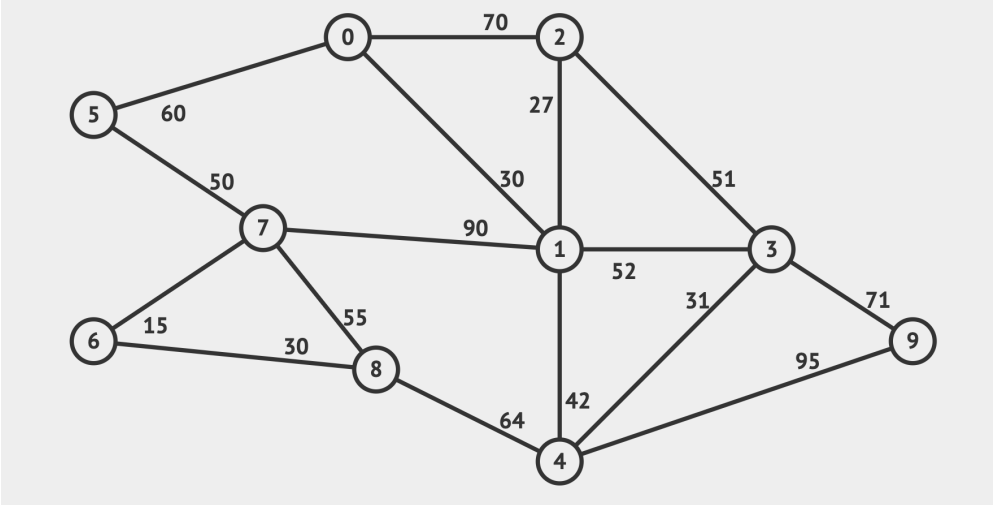
IV. HEAP

Given a list of keys [89, 40, 46, 55, 54, 5, 50, 73, 23, 47]

1) Present steps to build up a Binary Min/Max Heap
2) Iteratively extract items from the Heap above.
3) (*optional*) Using `java.util.PriorityQueue<>` class to build up a heap of integers in which
    a. Even numbers have higher priority than odd ones
    b. Among even numbers, larger integers have higher priority
    c. Among odd numbers, smaller integers have higher priority

V. GRAPH TRAVERSAL

Given the graph below

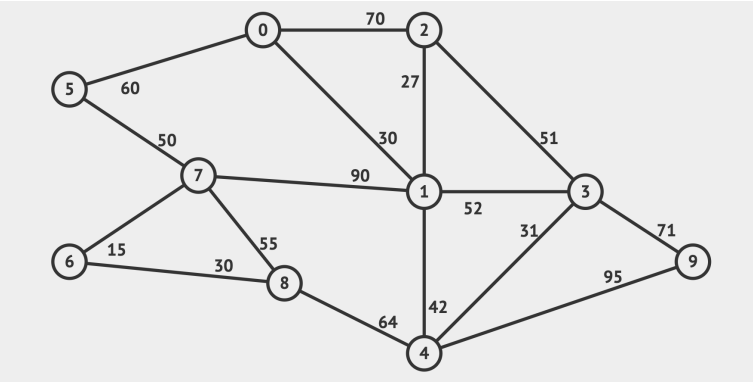


For each algorithm, including BFS and DFS,

- Perform the algorithm, starting from (0)
- Write down the list of keys in traversal order

If a vertex has several neighbors, then select the neighbor with the lower key to handle first.

VI. MINIMUM SPANNING TREE

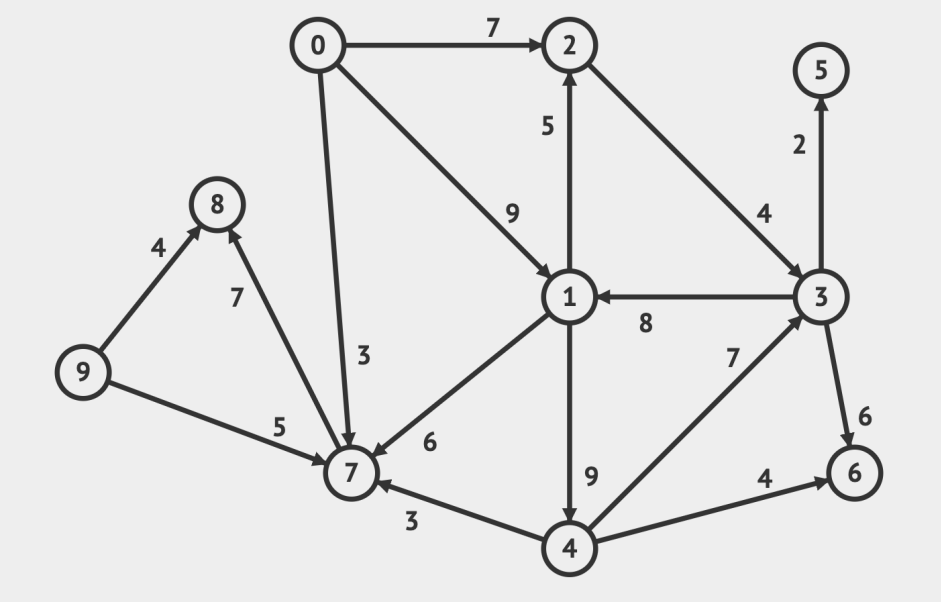


For each algorithm, including Prim's and Kruskal's

- Perform the algorithm
- Draw the final minimum spanning tree
- Write down the total cost of the minimum spanning tree

VII. SINGLE-SOURCE SHORTEST PATHS

Given the directed graph below



For each algorithm, including Bellman Ford's and Dijkstra,

- Perform the algorithm to find the shortest path from vertex 0 to the others
- Write down the path results and the corresponding cost.

Notes

- *For Bellman Fords, edges are handled in the ascending order of source vertices and destination ones. For instance,*

source (u)	*destination (v)*	*weight (w)*
0	1	2
0	3	4
1	2	6
1	4	2