CS302 – Data Structures using C++

Study Guide for the Final Exam Fall 2019

Revision 2.1

This document serves to help you prepare towards the final exam for the Fall 2019 semester.

1. What topics are to be covered by the final exam?

The final exam necessarily covers all topics discussed in class. However, certain topics will be prioritized. Those in particular relate to:

- Section 3: Stacks
- Section 6: Sorting Algorithms
- Section 8: Queues
- Section 9: Binary Search Trees
- Section 10: Heaps
- Section 11: Dictionaries
- Section 12: Balanced Search Trees | Red-Black Trees
- Section 13: Graphs
- Section 15: Using STL (specifically: for Heaps)

Those Sections should not be considered as "more important". It is just a selection for the exam.

2. What topics are most relevant for exam exercises involving coding?

The following are the most dominant – but not unique – sections from which coding exercises may be selected for the final exam:

- Section 6: Sorting Algorithms
- Section 8: Queues
- Section 9: Binary Search Trees
- Section 10: Heaps
- Section 11: Dictionaries
- Section 12: Red-Black Trees
- Section 13: Graphs
- Section 15: Using STL (specifically: for Heaps)

Those Sections should not be considered as "more important". It is just a selection for the exam.

2. How should I study for the final?

You are expected to be knowledgeable about:

- All what is discussed inside the class and captured by the online slides.
- All the material developed and designed in the framework of your homework Assignments #1-#4.
- All the material captured by the Quizzes inside the class.

Beyond this itemized list, it is important to highlight that:

- You are expected to answer theoretical questions, for example with respect to sorting algorithms efficiency.
- You are expected to be able to demonstrate understanding of how to design an abstract data type – that is to follow the design process outlined in the lectures of the class.

- You are expected to be able to code subsets of the implementation relevant to certain data types and algorithms. In the final you will not have to code on paper a particularly large piece of software. But you are definitely expected to be able to code subsets so prepare yourselves accordingly.

3. What code examples should I focus on?

You are expected to be familiar with:

- The code examples discussed inside the slides of the lectures and especially for those Sections discussed in Part 2 of this document.
- The code relevant to all your assignments.
- Further problem examples defined below <u>and others like that</u>
- Code examples related to problems uploaded on the course web page especially in the exam preparation section.

On Sorting Algorithms

1. Write an in-place merge sort algorithm that does not require a temporary array to merge the two halves. What is the efficiency of your solution?

2. You can sort a large array of integers that are in the range of 1 to 100 by using an array count of 100 items to count the number of occurrences of each integer in the array. Fill in the details of this sorting algorithm, which is called a bucket sort, and write a C++ function that implements it. What is the order of the bucket sort? Why is the bucket sort not useful as a general sorting algorithm?

On Queues

1. Use a queue to simulate the flow of customers through a check-out line in a store.

2. If you use a circular chain that has only a tail pointer, how do you access the data in the first node?

3. Define the method peekFront for the sorted list implementation of the ADT priority queue.

On Binary Trees

1. Provide definitions of the public method getNumberOfNodes and the protected helper method getNumberOfNodeshelper

2. Provide definition of the public method setRootData

3. Provide a definition of the public method getRootData. Recall that this method is a precondition.

4. Define the protected method postorder

5. Broadly be able to write down all discussed methods for tree traversal

6. Provide implementation of the addNode method (interface as in exercise 6)

7. Provide implementation of the method clear (interface as in exercise 6)

8. Provide implementation for the getHeight method (interface as in exercise 6)

9. Provide an implementation of the copyTree method

10. Provide an implementation of the destroyTree method using postorder traversal. What is the reasoning behind using postorder traversal?

11. Provide an implementation of the helper getHeightHelper using recursion.

12. Be able to derive the Big-O (computational complexity) for any step in the BST process.

On Binary Trees

1. Provide implementation for the Insertion method within a Red-Black Tree.

On Heaps

1. Use STL to make a heap consisting of 10 integers, add a new value that is the rounded mean of the numbers you just inserted, delete the maximum element in the heap and sort the heap.

2. On the heap definition based on Arrays, provide a method to return the index of the left child of a node.

3. On the heap definition based on Arrays, provide a constructor method.

4. On the heap definition based on Arrays, provide a method to peek its top.

On Dictionaries

1. Using the ADT dictionary operations, write pseudocode for a replace function at the client level that replaces the dictionary entry whose search key is x with another entry whose search key is also x.

2. Provide the definition of the method traverse for the Array Dictionary

3. Be able to implement a hashing function with or without conflict resolution.

On Graphs

1. Provide an implementation of the Dijkstra algorithm
2. Provide an implementation of the Breadth-First Search or Depth-First Search
3. Provide an implementation of the Minimum Spanning Tree algorithm

4. Provide a brute-force or simple heuristic (which will be detailed by us in the exercise declaration text in your exam) for the Traveling Salesman Problem

4. What are some indicative theoretical/design questions I should be able to respond to?

Below is an indicative set.

On Recursion

1. Given an integer n>0, write a recursive function countdown that writes the integers n, n-1, …, 1.

2. Write a program that uses recursion to solve the Towers of Hanoi problem.

3. Write the postfix expression that represents the following infix expression: (a*b – c)/d + (e-f)

4. What is the cost of the Towers of Hanoi with respect to the amount of moving operations?

5. Consider the following recursive function

```
int p(int x)
{
      if (x <= 3)
            return x;
      else
            return p(x-1) * p(x-3);
} // end p
```

Let m(x) be the number of multiplication operations that the execution of p(x) performs.

a. Write a recursive definition of m(x)
b. Prove that your answer to the above question is correct by using mathematical induction

On Stacks

1. Write a function that uses a stack to test whether a given string is a palindrome.

2. Design and implement a class of postfix calculators. Use the algorithm given in class to evaluate postfix expressions as entered into the calculator. Use only the operators +, -, *, and /. Assume that the postfix expressions are syntactically correct.

On Sorting Algorithms

1. Show that the merge sort algorithm satisfies the four criteria of recursion.

2. Trace a quick sort’s partitioning algorithm as it partitions the following array: 38 16 40 39 12 27

On Sorted Lists

1. Write specifications for the operation insertSorted when the sorted list must not contain duplicate entries.

2. Describe the concept of containment in the framework of a SortedListHasA being composed of an instance of the class LinkedList.

On Queues

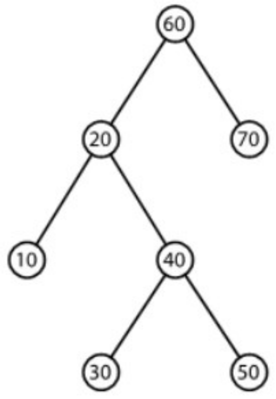
1. For each of the following situations, which of these ADTs (1 through 6) would be most appropriate? (1) a queue; (2) a stack; (3) a list; (4) a sorted list; (5) a priority queue; (6) none of these. If more than one apply, write one of them.
a. The customers at a deli counter who take numbers to mark their turn
b. An alphabetic list of names
c. Integers that need to be sorted
d. The boxes in a box trace of a recursive function
e. A grocery list ordered by the occurrence of the items in the store
f. The items on a cash register tape
g. A word processor that allows you to correct typing errors by using the Backspace key
h. A program that uses backtracking
i. A list of ideas in chronological order
j. Airplanes that stack above a busy airport, waiting to land
k. People who are put on hold when they call for customer service
l. An employer who fires the most recently hired person

On Trees and Binary Trees

1. Beginning with an empty binary search tree, what binary search tree is formed when you add the following letters in the order given? J, N, B, A, W, E, T

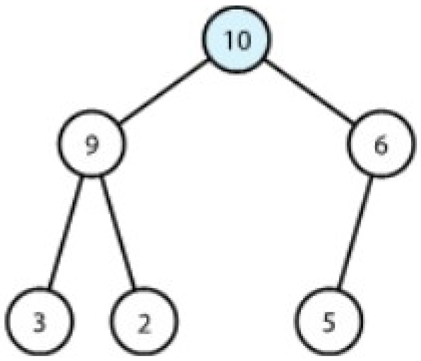
2. Arrange nodes that contain the letters A, C, E, F, L, V, and Z into two binary search trees: one that has minimum height and one that has maximum height.

3. Represent the following binary tree with an array



On Heaps

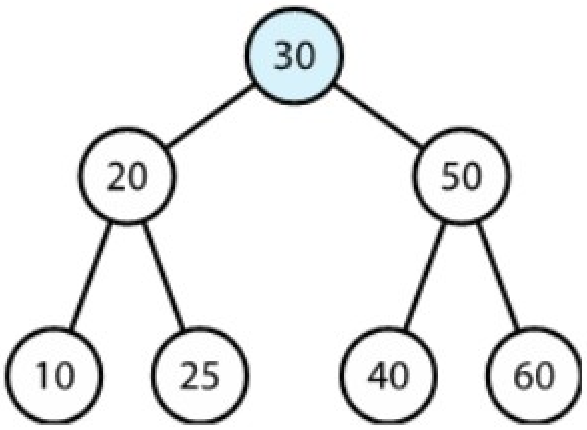
1. What array represents the maxheap shown below?



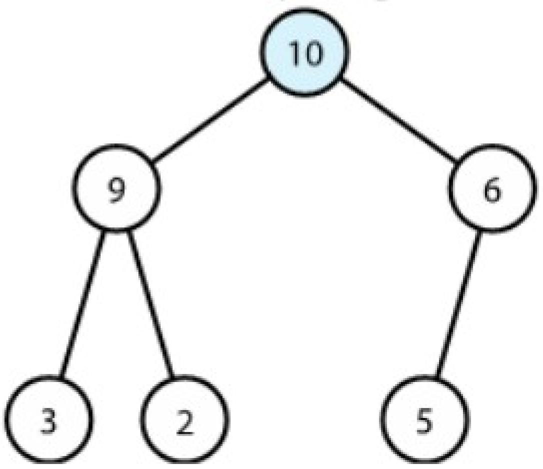
2. What complete binary tree does the following array represent?

5	1	2	8	6	10	3	9	4	7
0	1	2	3	4	5	6	7	8	9

3. is the full binary tree in the figure below, a semiheap?



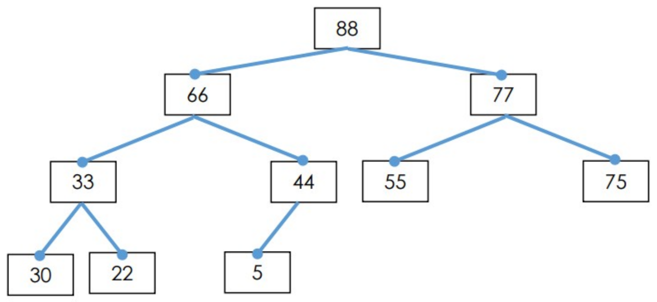
4. Consider the maxheap in the figure below, draw the heap after you add 12 and remove 12.



5. Visualize the initially empty myHeap after the following sequence of operations

```
myHeap.add(2)
myHeap.add(3)
myHeap.add(4)
myHeap.add(1)
myHeap.add(9)
myHeap.remove()
myHeap.add(7)
myHeap.add(6)
myHeap.remove()
myHeap.add(5)
```

6. Draw the following heap as an array



On Dictionaries

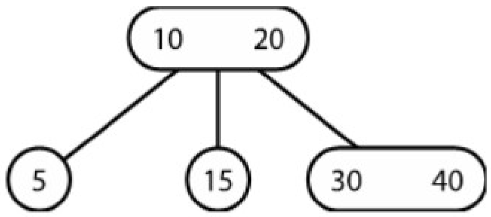
1. What is the Big-O function for addition, removal, retrieval and traversal for the case of:
a) unsorted array-based dictionary
b) unsorted link-based dictionary
c) sorted array-based dictionary
d) sorted link-based dictionary
e) BST-based dictionary

2. Compare implementations of the ADT Dictionary (with respect to adding an entry and removing). Comment about the unsorted array-based, sorted array-based, unsorted link-based and sorted link-based implementations.

On Balanced Search Trees

1. What is the result of adding 5, 40, 10, 20, 15, and 30 –in the order given- to an initially empty 2-3 tree?

2. What is the result of adding 3 and 4 to the 2-3 tree shown below?



3. Why does a node in a red-black tree require less memory than a node in a 2-3-4 tree?

4. Why can't a Red-Black Tree have a black child node with exactly one black child and no red child?

5. What is the maximum height of a Red-Black Tree with 14 nodes?

6. What is the main reason of using Red-Black Trees instead of conventional BSTs?

7. What color must the leaves be in a Red-Black Tree?

8. In a Red-Black Tree, if a node is red, what color are its children?

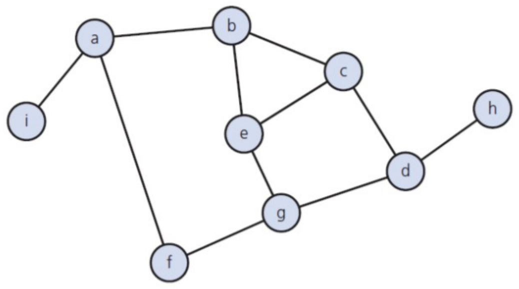
9. Derive the Big-O notation for Red-Black Trees

10. What is the **expected** insertion time for a Red-Black Tree

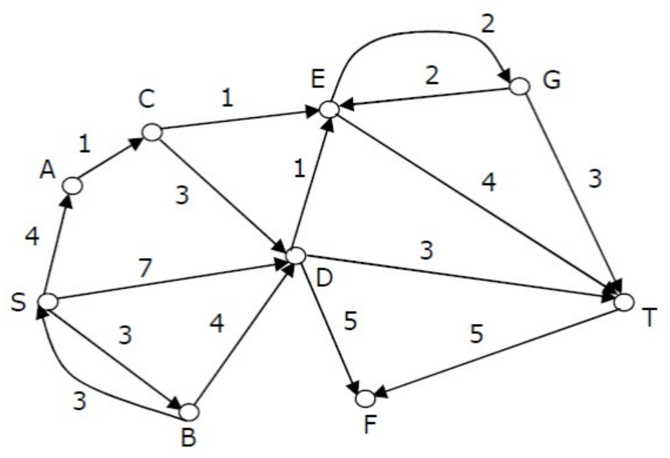
11. What are the differences, possible benefits and limitations, of Left-leaning Red-Black Trees.

On graphs

1. Write the adjacency matrix (or adjacency list) of the following graph

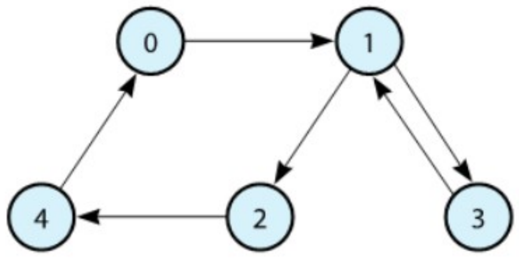


2. Consider the directed graph shown in the figure below. There are multiple shortest paths between vertices S and T. Which one will be reported by Dijstra's shortest path algorithm? Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered.



3. Is it possible for a connected undirected graph with five vertices and four edges to contain a simple cycle? Explain your answer.

4. Write the adjacency matrix and the adjacency list for the graph depicted below.

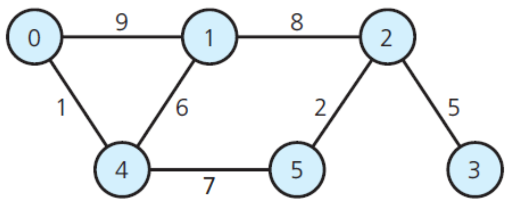


5. Use the depth-first strategy to traverse the graph in the Figure above, beginning with vertex 0. List the vertiess in the order visited.

6. Use the breadth-first strategy to traverse the graph in the Figure above, beginning with vertex 0. List the vertices in the order visited.

7. Is it possible for a connected undirected graph with five vertices and four edges to contain a simple cycle?

8. Draw the BFS spanning tree whose root is vertex 0 for the graph shown below.



9. Define the Traveling Salesman Problem