CS234 Fall 2017 Assignment 4
Due at 4:00 PM on Friday, December 1
Submit using MarkUs

Note: For all programming questions, you must use Python 3.2.3 or higher. Consider using the design recipe and Python style guide available from the course website. Approximately 80% of the marks of each question will be allocated to correctness, with the remaining 20% to style, including aspects of the design recipe. Examples and tests are optional.

Programming Component

1. (20 marks) Write a Python function get_successor to find the successor of a given node in a binary search tree. Recall: successor of a given node is defined as the node with the smallest key greater than the key of the given node. The input is the root node bstnode and the output should be a tree node or None.

Each tree node is defined as:
```python
class Node:
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None
        self.parent = None
```

NOTE:
1) You should not perform an inorder traversal of the binary search tree.
2) The sample code shown in the lecture is the solution for without parent links.
Written Component

It is highly recommended that you use computer software to finish the drawings. If you are going to submit hand drawings, please ensure that your drawings are readable. Otherwise, you may lose marks from unclear drawings.

2. (20 marks, 5 marks for each)
   For the following integer sequence, please use the insertion algorithms that are provided in the lectures to

   5 10 1 9 7 3 8 2 6 4

   a) Draw a **min-heap** that would be built from the sequence.
   b) Draw a **max-heap** that would be built from the sequence.
   c) Draw a **binary search tree** that would be built from the sequence.
   d) Draw an **AVL tree** that would be built from the sequence.

   **NOTE:** there is no partial mark for this question. For each of the drawings, you will receive either 0 or 5 marks. Please ensure your drawings are exactly what you expect before you submit.

3. (10 marks) Sort the above sequence in **increasing order** using in-place **HeapSort**. You must use the **max-heap** built in the above question.
   a) Show the array representation of the max-heap you built. You don’t need to show the steps. Only the array content is enough.
   b) Show the steps of extracting the items in-place and obtaining the sorted list.

   **NOTE:**
   1) You must show both the heap and its array representation for each step as described in the course slides.
   2) In each step, you must clearly indicate the part of the array representing the sorted sequence of items.
   3) In each step, you only need to show the heap and its array representation for two statuses: (i) when an item is extracted from the heap (with an appropriate node taking its position) (ii) its final representation of the heap after sift-downs. You don’t need to show all the steps of sift-down.
The following example is showing the first two steps of building up the max-heap, which is how you may achieve the result of Question 3a). You may use the same format to show your extraction steps in Question 3b).

Step 1.
(i) Insert 5

(\[
\begin{array}{cccccccc}
5 & 10 & 1 & 9 & 7 & 3 & 8 & 2 & 6 & 4 \\
\end{array}
\] )

(ii) Sift-up 5

(\[
\begin{array}{cccccccc}
5 & 10 & 1 & 9 & 7 & 3 & 8 & 2 & 6 & 4 \\
\end{array}
\] )

Step 2.
(i) Insert 10

(\[
\begin{array}{cccccccc}
10 & 5 & 1 & 9 & 7 & 3 & 8 & 2 & 6 & 4 \\
\end{array}
\] )

(ii) Sift-up 10

(\[
\begin{array}{cccccccc}
10 & 5 & 1 & 9 & 7 & 3 & 8 & 2 & 6 & 4 \\
\end{array}
\] )

4. (15 marks) Consider the following graph

a) Show the adjacency matrix for this graph.

b) Illustrate the performance of Dijkstra’s algorithm on this graph for finding the shortest path from vertex V1 to all other vertices in the graph.

**NOTE:** At each iteration of the algorithm, you must clearly show the length of the current shortest-known path from V1 to all the other vertices. Illustrate the steps as we did in class by showing a series of tables