American University of Armenia, CSE
CS121 Data Structures A, C
Fall 2021

Homework Assignment 7

Due Date: Thursday, December 9 by 23:59 electronically on Moodle

Please solve the programming tasks either in Java or C++, following good coding practices (details are on Moodle).

1. **(19 points)** Write a class `UnsortedSLLMap` that extends the `AbstractMap` class using a singly linked list as the underlying data structure. However, you shouldn't use the `SinglyLinkedList` class; rather you should have instance variables for the head and tail nodes, the map size. You will also need a nested `Node` class.

   Your class should support all of the following functionality:

   (a) two constructors that create an empty map: a no-arg constructor that relies on the default comparator, and another constructor that receives a comparator argument for the key-type;

   (b) `size` method (note that `isEmpty` is inherited from `AbstractMap`);

   (c) the fundamental methods `get`, `put`, and `remove` from the Map ADT;

   (d) the `entrySet` method from the Map ADT (note that `keySet` and `values` methods are inherited from `AbstractMap`).

   You may want to add utility methods to your class to simplify the implementation of the above functionality.

2. **(38 points)** Write a class `BSTMap` that extends the `AbstractSortedMap` class using a `LinkedBinaryTree` *tree* as the underlying data structure. Note that it should maintain *tree* as a binary search tree. All the external nodes in a BST are sentinel nodes, i.e. they store `null` instead of a real entry. This implies that when the map is empty, the *tree* BST should contain only a single sentinel node at the root.

   Your class should support all of the following functionality:

   (a) two constructors that create an empty tree: a no-arg constructor that relies on the default comparator, and another constructor that receives a comparator argument for the key-type;

   (b) `size` method (note that `isEmpty` is inherited from `AbstractMap`);

   (c) a utility method `expandExternal`;

   (d) a utility method `treeSearch`;

   (e) the fundamental methods `get`, `put`, and `remove` from the Map ADT;

   (f) the `firstEntry, lastEntry, floorEntry, ceilingEntry, lowerEntry`, and the `higherEntry` methods from the Sorted Map ADT;

   (g) the `entrySet` method from the Map ADT (note that `keySet` and `values` methods are inherited from `AbstractMap`);

(h) the `subMap` method from the Sorted Map ADT.

You may want to add more utility methods to your class to simplify the implementation of the above functionality.

3. (14 points) Write an efficient method that, given the root node of a binary search tree (this is the `Node` class for linked binary trees), determines the maximum difference of the heights of the children among all internal nodes in the tree. The entries have integer keys and integer values.

4. (14 points) Write a generic method that, given the root node (this is the `Node` class for linked binary trees) of a BST or AVL tree, returns the key of the median entry.

5. (15 points) *A problem on hash tables will be released separately.*

5. (15 points) Consider the following set of words and illustrate the results as demanded in 1) and 2):

december, finals, holidays, grade, BST, DS, pass, hash, sash, shah, sentinel, gift, fun, happy, me

1) Given the words above, insert them one-by-one in the order given into a hash table of size N = 13 using exclusive-or (by their ASCII decimal codes) as your hash code and modular division as your compression function. If collisions occur, you may handle them by using separate chaining.

2) Given the words above, insert them one-by-one in the order given into a hash table of size N=19 using polynomial accumulation (using ASCII decimal codes) as your hash code and MAD as your compression function. If collisions occur, you may handle them by using linear probing.

    a. For the stated problem above use a=33
    b. For the stated problem above use a=31
    c. For the stated problem above with a = 33, n = 15, N=19 calculate the number of steps for get("J") in the table.
    d. For the stated problem above with a = 31, n = 15, N=19 calculate the number of steps for get("J") in the table.

For **ASCII decimal codes** you may refer to http://www.asciitable.com/