CS 202, Fall 2019

Homework #4 – Balanced Search Trees, Hash Tables and Graphs

Due Date: December 25, 2019

Important Notes

Please do not start the assignment before reading these notes.

- Before 23:55, December 25, upload your solutions in a single **ZIP** archive using Moodle submission form. Name the file as `studentID_hw4.zip`.
- Your ZIP archive should contain the following files:
  - `hw1.pdf`, the file containing the answers to Questions 1, 2 and 3.
  - `openHashTable.cpp, openHashTable.h, chainHashTable.cpp, chainHashTable.h, main.cpp` and `any other files` which contain the C++ source codes, and the `Makefile`.
  - Do not forget to put your name, student id, and section number in all of these files. Well comment your implementation. Add a header as in Listing 1 to the beginning of each file:

Listing 1: Header style

```
/**
 * Title: Balanced Search Trees, Hash Tables & Graphs
 * Author: Name Surname
 * ID: 21000000
 * Section: 1
 * Assignment: 4
 * Description: description of your code
 */
```

  - Do not put any unnecessary files such as the auxiliary files generated from your favorite IDE. Be careful to avoid using any OS dependent utilities (for example to measure the time).

  - You should prepare the answers of Questions 1, 2, 3 using a word processor (in other words, do not submit images of handwritten answers).
  - Use the exact algorithms shown in lectures.

- Although you may use any platform or any operating system to implement your algorithms and obtain your experimental results, your code should work on the **dijkstra** server (dijkstra.ug.bcc.bilkent.edu.tr). We will compile and test your programs on that server. Please make sure that you are aware of the homework grading policy that is explained in the rubric for homeworks.

- This homework will be graded by your TA, Mubashira Zaman. Thus, please contact her directly for any homework related questions.

Attention: For this assignment, you are allowed to use the codes given in our textbook and/or our lecture slides. However, you ARE NOT ALLOWED to use any codes from other sources (including the codes given in other textbooks, found on the Internet, belonging to your classmates, etc.). Furthermore, you ARE NOT ALLOWED to use any data structure or algorithm related function from the C++ standard template library (STL).

Do not forget that plagiarism and cheating will be heavily punished. Please do the homework yourself.

Question 1 – 10 points

(a) [*3 points*] What is the maximum height of a red-black tree with n nodes?

(b) [*3 points*] Which of these has more space complexity: AVL trees or red-black trees? Give a 1-line reason for your choice.

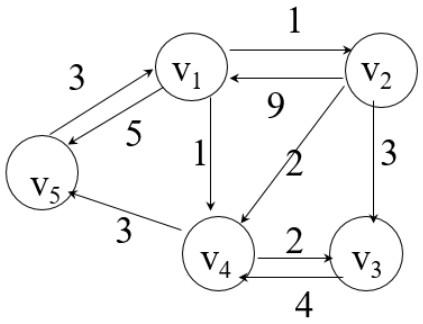
(c) [*4 points*] When finding the minimum spanning tree of a graph, what is the stopping criteria in Prim's algorithm?

Question 2 – 20 points

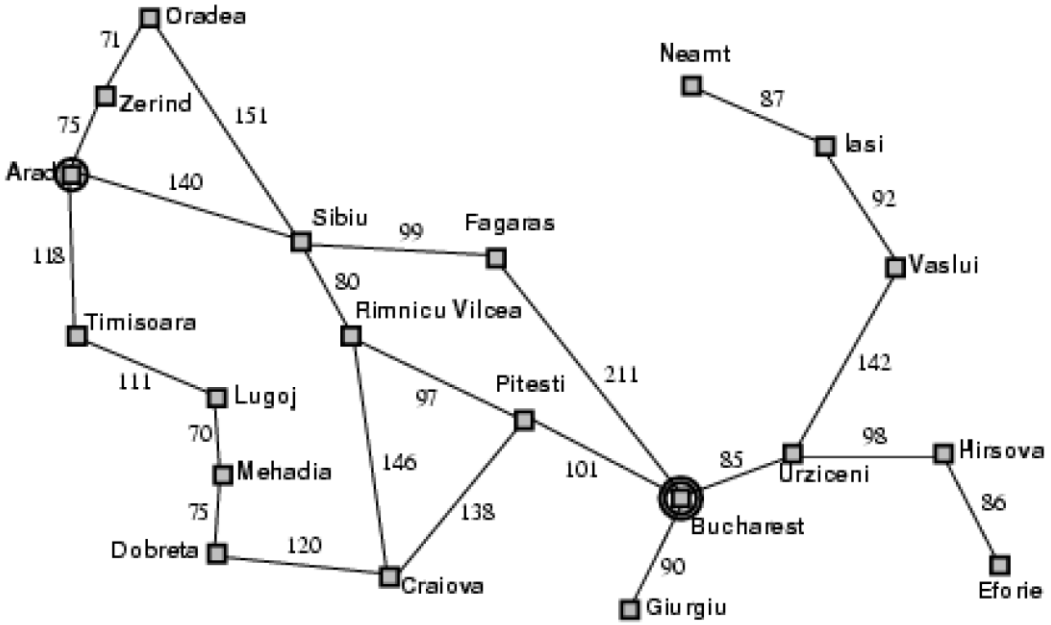
(a) [*5 points*] Insert items with the following key values in a 2-3-4 tree: [36, 39, 40, 32, 20, 36, 21, 26, 28, 38, 50, 37, 44, 41]. Show only the final tree.

(b) [*5 points*] Convert the previous 2-3-4 tree to a red-black-tree.

(c) [*5 points*] Find the shortest path from v_5 to v_3 using Dijkstra's shortest path algorithm. Draw the trace table as well.



(d) [*5 points*] A city map of Romania is given below. Find the path and the length of the path from Arad to Bucharest using Breadth First Search. Make sure to show your work.



Question 3 – 70 points

In this question, you are asked to develop an inventory for a fruits and vegetables storage warehouse using hash tables. Each row in the file "items.txt" shows the amount of items available and their name. Implement two hash table classes. One is for open addressing with quadratic probing called `OpenHashTable`. The second class will implement separate chaining and will be called `ChainHashTable`. We expect no more than 50 entries at the same time in each table. Both classes must have the following functionalities:

(a) [*20 points*] Create a hash function by taking the item name as input and generate a key for it. You will choose the appropriate hashing function and the size of the hash table yourself. In the pdf report, mention the hashing function and the size of the hash table that you choose, and give your reasoning for choosing them. Where applicable, make sure to maintain status of each location as empty, occupied, and deleted.

(b) [*10 points*] Take the name and amount of item as input and insert it into the hash table using the key generated from the hash function. Both the item name and amount must be stored in the hash table entry.

(c) [*10 points*] Find an item by searching its name in the hash table and print its name and its available quantity.

(d) [*10 points*] Delete an item from the hash table.

(e) [*5 points*] Print the entire hash table in the following manner:

```
0: Apple = 3567,
1: Mango = 463,
2: Guava = 4653,
3: Banana = 1569,
.
.
.
```

For separate chaining, print the items stored at one location in the same line.

(f) [*15 points*] In this part, you will analyze the performance of the two hashing classes. Your function will find the average time spent for insertion of all items from the text file, the average retrieval time of 100 random items and the average deletion time of 100 random items. For both the hash tables, find the number of comparisons made for lookup in the retrieval operation. Also find the number of collisions occurring in insertion for the OpenHashTable. Your function should print an output as follows:

Listing 2: Sample output

```
-----------------------------------------------------
Part a - Analysis of Open Addressing Hash Table
Parameter                    Time Elapsed
Average Insertion Time           x ms
Average Retrieval Time           x ms
Average Deletion Time            x ms

Number of comparisons in Retrieval = y
Number of collisions in Insertion = y
-----------------------------------------------------
Part b - Analysis of Separate Chain Hash Table
Parameter                    Time Elapsed
Average Insertion Time           x ms
Average Retrieval Time           x ms
Average Deletion Time            x ms

Number of comparisons in Retrieval = y
-----------------------------------------------------
```

Plot a `bar chart` for these values and include it in your pdf report.

(g) [*mandatory*] Create a `main.cpp` file which does the following in order: (repeat the following steps for both the hash table classes; first for OpenHashTable and then for ChainHashTable).

- Read the input items.txt file line by line and inserts all the rows in the hash table.
- Print the hash table.
- Find your favorite 3 fruits.
- Delete your 3 least favorite vegetables.
- Print the hash table.

The prototypes of the functions are relaxed and can be changed if needed. If needed, you can create other cpp code files that support your code.

At the end, write a basic Makefile which compiles all of your code and creates an executable file named `hw4`. Check out these tutorials for writing a simple make file: tutorial 1, tutorial 2. Please make sure that your `Makefile` works properly, otherwise you will not get any points from this question.