CS 202 - Spring 2022
Homework 4
Due: 23:55, May 11, 2022

Important Notes

Please do not start the assignment before reading all the notes.

1. Before 23:55, May 11, upload your solutions in a single **ZIP** archive using Moodle submission form. Name the file as `<studentID>_<secNo>_hw4.zip`.

2. Your ZIP archive should contain the following:

   - `hw4.pdf`, the file containing the answers to Questions 1 and 2 and the report of Question 3.
   - `q3`, the directory containing all the source files and the `Makefile` for Question 3.

3. Do not forget to put your name, student id, and section number in all of these files. Properly comment your implementation. Add a header as given below to the beginning of each file:

```
/*
 * Title: CS202 Spring 2022
 * Author: Name Surname
 * ID: 22000000
 * Section: 9
 * Assignment: 4
 * Description: description of your code
 */
```

4. Do not put any unnecessary files such as the auxiliary files generated from your favorite IDE.

5. Be careful to avoid using any OS-dependent utilities.

6. You should upload handwritten answers for Questions 1 and 2 (in other words, do not submit answers prepared using a word processor).

7. Use the exact algorithms shown in lectures.

8. 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. Thus, you will lose significant points if your C++ code does not compile or execute on the `dijkstra` server.

9. This homework will be graded by your TA, Batuhan Kaynak. Thus, please contact him directly (batuhan.kaynak -at- bilkent.edu.tr) 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.), or data structures and algorithms 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) (5 points) Starting with an empty 2-3 tree, perform the following operations on the tree in the given order:

- Insert 2
- Insert 20
- Insert 6
- Insert 16
- Insert 10
- Delete 2
- Insert 12
- Insert 14
- Insert 8
- Delete 16
- Insert 18
- Insert 3
- Delete 6
- Delete 14

Show the underlying tree after each insertion and deletion.

(b) (5 points) Repeat part (a) for a 2-3-4 tree.

Question 2 (15 points)

(a) (5 points) Assume that we have a hash table with size 13 (index range is $0 \ldots 12$), and we use

$$h(x) = x \mod 13$$

as the hash function to map a given numeric key into this hash table. Assuming *open addressing with linear probing*, draw the hash table after the insertion of the keys

`45 64 54 17 69 58 32 60 26`

in the given order.

Find the average number of probes for a successful search and an unsuccessful search for the final hash table.

(b) (5 points) Repeat part (a) for *open addressing with quadratic probing*.

(c) (5 points) Repeat part (a) for *separate chaining*.

Question 3 (75 points)

(a) You are asked to develop a driver program that allows an airline company to perform some operations on a flight map.

- The flight map is represented as a weighted undirected graph where the vertices correspond to the airports and the edges correspond to the flights between these airports.
- The weight of each edge represents the duration of the flight.
- The operations to the program execution are provided by an input file.
- The first line of the input file contains an integer n, which is the number of airports.
- Each airport is denoted by an integer from 0 to $n-1$, inclusive.
- The second line of the input file contains an integer m, which is the number of operations.
- The following m lines are operations in the following format:
  `<opCode> <arguments>`
  - `opCode` is a single character operation code, matching one of the operations below.
  - `arguments` is a single line of arguments for the related operation.
- Possible operations:
  - **Insertion:**
    opCode: `I`
    arguments: u v w
    Details: Inserts a two-way flight between two airports u and v. w is the duration of the flight, represented as an integer. If there is a previous flight between u and v, keep both.
    Expected output:

```
Inserted a new flight between <u> and <v>.
  The number of flights from <u> is <x>.
```

  - **List:**
    opCode: `L`
    arguments: u
    Details: Lists the flights from airport u.
    Expected output:

```
List of flights from <u>:
  <u> to <v1> with a duration of <w1>.
  <u> to <v2> with a duration of <w2>.
  ...
  The number of flights is <xx>.
```

– **Shortest Path:**
opCode: `S`
arguments: *s t*
Details: Reports the shortest possible path from airport *s* to airport *t*.
Expected output:

```
The shortest path from <s> to <t>:
  <s> to <v1> with a duration <w1>
  <v1> to <v2> with a duration <w2>
  ...
  <vn-2> to <vn-1> with a duration <wn-1>
  <vn-1> to <t> with a duration <wn>
  Total flight duration of path: <xx>
```

or, if there is no path from *s* to *t*:

```
No paths from <s> to <t>.
```

– **Minimize Costs:**
opCode: `M`
arguments: *no arguments*
Details: The company requests a cost minimization. Assume that the cost of a two-way flight is equal to the duration. To minimize, they want to cancel some of the flights, while also keeping the connectivity. If two airports are reachable before the operation, they should also be reachable after the operation. This operation should lead to the minimum possible cost under the constraints. This operation should modify the underlying graph.
Expected output:

```
Total cost of operations before minimization: <xx>
Total cost of operations after minimization: <yy>
```

- The program should execute the operations one by one.
- The underlying graph should be an adjacency list implementation.
- You should provide an efficient implementation for each operation.
- The name of the executable of your program should be `hw4`.
- Your program should accept the path to the input file as a parameter to your program, e.g. if the input file is `in.txt` in the same directory as the executable, then the following command should execute your program:

```
hw4 in.txt
```

Example Input File

```
5
12
I 0 1 2
I 0 2 3
I 0 4 6
I 1 2 6
S 2 3
I 3 4 3
S 2 3
I 1 3 6
L 2
S 2 3
M
L 2
```

Example Output

```
Inserted a new flight between 0 and 1.
  The number of flights from 0 is 1.
Inserted a new flight between 0 and 2.
  The number of flights from 0 is 2.
Inserted a new flight between 0 and 4.
  The number of flights from 0 is 3.
Inserted a new flight between 1 and 2.
  The number of flights from 1 is 2.
No paths from 2 to 3.
Inserted a new flight between 3 and 4.
  The number of flights from 3 is 1.
Inserted a new flight between 1 and 3.
  The number of flights from 1 is 3.
List of flights from 2:
  2 to 1 with a duration of 6.
  2 to 0 with a duration of 3.
  The number of flights is 2.
The shortest path from 2 to 3:
  2 to 0 with a duration 3
  0 to 1 with a duration 2
  1 to 3 with a duration 6
  Total flight duration of path: 11
Total cost of operations before minimization: 26
Total cost of operations after minimization: 14
List of flights from 2:
  2 to 0 with a duration of 3.
  The number of flights is 1.
```

(b) Prepare a report and discuss the worst-case asymptotic complexities of your implementation of operations.