CS202, Fall 2024

Homework 3 - Hashing

Due: 23:59, 13/12/2024

Before you start your homework please read the following instructions carefully:

FAILURE TO FULFIL ANY OF THE FOLLOWING REQUIREMENTS WILL RESULT IN A GRADE SCORE OF 0 (zero) WITHOUT ANY CHANCE OF REDEMPTION.

- See the course page for any late submission policies and Honor Code for Assignments.
- Upload your solutions in a single ZIP archive using the Moodle submission form. Name the file as studentID_name_surname_hw3.zip.
- Your ZIP archive should contain **only** the following files:
  - **studentID_name_surname_hw3.pdf**, the file containing the answers to Questions 6, 7, and 8.
  - In this assignment, you must have separate interface and implementation files (i.e., separate .h
  - and .cpp files) for your class. Your class name MUST BE **HashTable** and your file names MUST BE
  - **HashTable.h** and **HashTable.cpp**. You have to implement **substrings.cpp**, **minimalset.cpp** ,**documents.cpp, lexicographic.cpp,** and **insertionorders.cpp** to solve and test Question 1-5. Note that you may write additional class(es) in your solution.
  - Your .cpp, .h , .pdf files, and **Makefile** , which produces executables for each question (No Makefile or broken Makefile results in 50% deduction in points).
  - Do not forget to put your name, student id, and section number in all of these files. Comment your implementation well. Add a header (see below) to the beginning of each file:

```
/**
 * Title: Hashing
 * Author : Name & Surname
 * ID: 12345678
 * Section : 1
 * Homework : 3
 * Description : description of your code
 */
```

  - Do not add unnecessary files such as the auxiliary files generated from your preferred IDE.
  - Please do not use **Turkish** letters in your file and folder names.
- Your code must compile.
- Your code must be complete.
- You ARE NOT ALLOWED to use any data structure or algorithm-related function from the C++ standard template library (STL) or any other external libraries.
- We will test your code using the **Google test library**. Therefore, the output message for each operation in Questions 1-5 MUST match the format shown in the output of the example code.
- Your code must run on the **dijkstra.cs.bilkent.edu.tr** server.
- For any questions related to the homework, contact your TA: *ziya.ozgul@bilkent.edu.tr* or you can ask your questions on forum.

Question 1 - 20 points

You are given one text, length of n, and m unique patterns. You want occurrences of each pattern in the text. You are given text in the first line. The number of patterns is given in the second line. Patterns are provided afterwards. The shortest and the longest pattern length differ at most by 5. Assume all texts and patterns contain only a-b-c-d.

```
Sample Input( patterns.txt )
abcbccdaaabcdbccd
4
bc
ab
cd
bcc
Sample Output( occurrences.txt)

4
2
3
2
```

Explanation:

First pattern, “bc”, occurs at 2,4, 11 and 14 indices(one based indexing).
Second pattern, “ab”, occurs at 1 and 10.
Third pattern, “cd”, occurs at 6,12 and 16.
Forth pattern, “bcc”, occurs at 4 and 14.

- This subtask MUST be done using hashing.
- To get a full credit your code must work correctly and work in $O((N + M) * log N + total\,number\,of\,characters)$ and within 1-2 seconds per input.
- If you try to solve this question using data structures other than hash tables or hashing you will get 0 from this part and possibly an FZ from this course depending on your answers to other questions.
-The name of the input file will be provided as a command line argument to your program. Thus, your program should run using two command line arguments. Therefore, the application interface is simple and given as follows:

Hint:

You may read the course slides and substring matching problems and Rabin Karp Algorithm. Use a rolling hash.

```
username@dijkstra:~>./substrings <input_filename> <output_filename>
```

Assuming that you have an executable called “substrings”, this command calls the executable with one command line argument, the name of the file from which your program reads the number of strings and strings.

Note that outputfile name is optional.

Question 2 - 20 points

You are given n unique strings. Strings are composed of lower-case letters. You can make two operations on strings. The first one is reverse operation. For example, the string “abcd” becomes “dcba” after the reverse operation. The second one is the cyclic shift operation. For example, the string “abcd” becomes “bcda” after shift, and “bcda” becomes “cdab”. However, there is a restriction. You have to complete all reverse operations before you start shift operations. By doing these operations, you can make strings equal. Your aim is to find the minimal subset of given strings. Minimal subset is defined as a subset of given strings with the minimum number of elements. In other words, when you reach a minimal subset you cannot minimize the strings further. For example, you have strings “abcd” and “bcda”. When you apply shift on first one you only have “bcda” and this is a minimal subset. Note that there can be different minimal subsets but the size of the minimal subset is unique. However, you are required to minimize the number of reverse operations when you make the given set minimal. On the other hand, there is no limitation to shift operations.

You have a number of strings in the first line and you have strings after that.

You are expected to print the number of reverse operations (first line of output) and size of minimal subset(second line).

```
Sample Input 1( strings.txt )
2
abcde
edcba
Sample Output( output.txt)
1
1
```

Explanation:

The size of the minimal subset is 1. Minimal subset can be either “abcde” or “edcba” and its size is 1. You need to have one reverse operation on “abcde” to make it equal to “edcba” or vice versa.

line).

Sample Input 2(strings.txt)
4
baaa
aaab
bcd
dcb
Sample Output(output.txt)
1
2

Explanation:

The size of the minimal subset is 2. Minimal subset can be {“baaa”, “bcd”} and its size is 2. You need to have one reverse operation on “bcd” to make it “dcb” or vice versa. “baaa” is the reverse of “aaab” but you don’t need reverse operation. When you apply shift on “baaa”, you obtain “aaab”.

- This subtask MUST be done using hashing.
- If you try to solve this question using data structures other than hash tables you will get 0 from this part and possibly an FZ from this course depending on your answers to other questions.
- To get a full credit your code must work correctly and work in O($total\ number\ of\ characters\ *\ log\ N$) (or better) and within 1-2 seconds per input.
-The name of the input file will be provided as a command line argument to your program. Thus, your program should run using two command line arguments. Thus, the application interface is simple and given as follows:

```
username@dijkstra:~>./minimalset <input_filename> <output_filename>
```

Assuming that you have an executable called “minimalset”, this command calls the executable with one command line argument, the name of the file from which your program reads the number of strings and strings.

Note that outputfile name is optional.

Important: The number of strings <= 100000 and the total number of characters in strings <= 10000000 in each test case for each subtask. You need to make your calculations based on this. You need to determine your modulus(hash table sizes) based on this to handle collisions.

Question 3 - 20 points

Ziya has a storage problem in his laptop. He has limited resources and wants to increase the available storage. Therefore, he wants to reduce the number of documents. He has an N documents. Some of his documents are similar to each other and he wants to encrypt them. His encoding method is that he converts similar documents to common documents with little changes. All of the documents are strings. He can add at most one new character to

document or delete at most one character from document or he can change at most one character in one document, he can do cyclic shifting on document. By doing this, he can decrypt the particular document from a common document when he needs to. He wants to minimize the number of common documents. If two different documents can be converted to the same document, they will have a common document in optimal solution. First line of input denotes the number of documents, N. Then, documents are listed.

Sample Input 1(strings.txt)
5
abc
acc
bca
xyz
xyzt
Sample Output(output.txt)
2

Explanation:
Minimum number of documents is 2. abc, acc and bca can be converted to abc. acc can be converted to abc by changing the second character. bca can be converted to abc by shifting operations. xyz can be converted to xyz by appending "t" at the end of it.

- This subtask MUST be done using hashing.
- If you try to solve this question using data structures other than hash tables you will get 0 from this part and possibly an FZ from this course depending on your answers to other questions.
- To get a full credit your code must work correctly and work in O($total\ number\ of\ characters\ *\ log\ (total\ number\ of\ characters)$) (or better) and within 1-2 seconds per input.
-The name of the input file will be provided as a command line argument to your program. Thus, your program should run using two command line arguments. Thus, the application interface is simple and given as follows:

```
username@dijkstra:~>./documents <input_filename> <output_filename>
```

Assuming that you have an executable called "documents", this command calls the executable with one command line argument, the name of the file from which your program reads the number of strings and strings.

Note that outputfile name is optional.

Important: The number of strings <= 100000 and total number of characters in strings <= 10000000 in each test case for each subtask. You need to make your calculations based on this. You need to determine your modulus(hash table sizes) based on this to handle collisions.

Question 4 - 20 points

Ziya is given a final version of hash table of size N, which is implemented using linear probing for collision resolution. The given hash table consists only of natural numbers and empty parts are denoted as -1. He wants to find the lexicographically smallest insertion order of elements to the hash table. The insertion sequence A is lexicographically smaller than B if the first differing index, call it i, A[i] < B[i]. If there is no possible insertion order to obtain a given hash table, the output should be “Impossible”.

```
Sample Input( hashtable.txt )

3
30 19 9
Sample Output( insertionorder.txt)
19 30 9
```

Explanation:
Possible orders to obtain this hash table are (30 19 9) or (19 30 9). (19 30 9) is lexicographically smaller than (30 19 9).
19 is added to the hash table, 19 % 3 = 1. So, it is placed on 1.
30 is added to the hash table, 30 % 3 = 0. So, it is placed on 0.
9 is added to the hash table, 9 % 3 = 0. There is a collision. It finds empty space on the 2nd index.

```
Sample Input( hashtable.txt )

2
1 0
Sample Output( insertionorder.txt)
Impossible
```

Important:

- This question MUST be solved using the linear probing method for collision resolution.

- If you try to solve this question using data structures other than hash tables you will get 0 from this part and possibly an FZ from this course depending on your answers to other questions.
- You can implement hash tables using the implementations in lecture slides.
- To get a full credit your code must work correctly and work in O(N^2) and within 1-2 seconds per input.

The name of the input file will be provided as a command line argument to your program. Thus, your program should run using two command line arguments. Thus, the application interface is simple and given as follows:

```
username@dijkstra:~>./lexicographic <input_filename> <output_filename>
```

Assuming that you have an executable called "lexicographic", this command calls the executable with one command line argument, the name of the file from which your program reads the number of strings and strings.

Note that outputfile name is optional.

Question 5 - 20 points

Ziya is given a final version of hash table of size N, which is implemented using linear probing for collision resolution. The given hash table consists only of natural numbers and empty parts are denoted as -1. He wonders the number of different insertion orders producing this table but he does not know how he can compute this. Ziya finds that he is able to insert any element in at most 3rd place. In other words, for any element X, in this hash table, there exists an insertion order such that this element is one of the first 3 inserted elements. However, he still doesn't know how to do it. Therefore, he wants you to find a number of possible insertion orders which give this final table. Since this number could be extremely large, you are expected to return it in mod 1000000009.

Sample Input(hashtable.txt)

```
3
30 19 9
```

Sample Output(numberoforders.txt)

```
2
```

Explanation:

Possible orders to obtain this hash table are (30 19 9) or (19 30 9). (19 30 9) is lexicographically smaller than (30 19 9).

19 is added to the hash table, 19 % 3 = 1. So, it is placed on 1.

30 is added to the hash table, 30 % 3 = 0. So, it is placed on 0.

9 is added to the hash table, 9 % 3 = 0. There is a collision. It finds empty space on the 2nd index.

Important:

- This question MUST be solved using the linear probing method for collision resolution.
- If you try to solve this question using data structures other than hash tables you will get 0 from this part and possibly an FZ from this course depending on your answers to the other questions.
- You can implement hash tables using the implementations in lecture slides.
- To get a full credit your code must work correct and work in O(N) and within 1-2 seconds per input.

The name of the input file will be provided as a command line argument to your program. Thus, your program should run using two command line arguments. Thus, the application interface is simple and given as follows:

```
username@dijkstra:~>./insertionorders <input_filename> <output_filename>
```

Assuming that you have an executable called "insertionorders", this command calls the executable with one command line argument, the name of the file from which your program reads the number of strings and strings.

Note that outputfile name is optional.

You will be graded based on your best 4 solutions for the first 5 questions.

Important Note for Outputs

You can add the output file name as an argument to print the outputs of your code.
When I grade your homeworks, I will also give the output file name as a parameter to your program.
Therefore, please add this script to your implementation for each subtask.

```
!!!!!
if ( argc > 2 ){
    freopen(argv[2], "w", stdout);
}
!!!!!
```

Important

- This homework requires you to use hashing.

- You are expected to solve the questions using either hash tables or hashing techniques like rolling hash.

- Think carefully about the questions.

- Executable names are very important. You must use executable names given above.

- Please write your makefiles carefully.

- Test your code on dijkstra.

- Follow forum regularly. Read the discussions on forum before asking questions. It may be asked already. If there is a same or closely related discussion to your question, ask your question under the existing topic. If there is no topic related to your question, open a new topic with a meaningful name. Misuse of forum may be penalized.

Question 6 - 5 points

What if there is no restriction on patterns in Question 1 ? How can you solve the question? Can you still solve this in given time complexity using hashing ?

Question 7 - 5 points

What is the minimum and maximum number of insertion orders in Question 5 if the size of hash table is N. What are the conditions for obtaining this?

Question 8 - 15 points

In this question, you will analyze your solutions for each question. You must discuss the question, discuss about the worst case scenarios. You also need to discuss your solution. You need to talk about your hash strategies. You are expected to mention hash functions. Also, you need to talk about how you handle collisions. What is the main advantage of hashing in these questions ? What have you done for the collision scenarios? You MUST NOT write this part in only 2-3 sentences. It should be detailed enough to get a full credit.