VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

ASSIGNMENT 2

JJK RESTAURANT OPERATIONS

(PART 2 - FLASHBACK)

HO CHI MINH CITY, NOVEMBER 2023

Assignment 2 Specification

Version 1.0

1 Outcomes

After completing the assignment, you will be able to:

- Implement tree structures and get familiar with hash table.
- Choose and apply appropriate structures to achieve the desire result.

2 Preface

Chào các giải thuật sư, do đã quá lâu mà tác giả vẫn chưa cho Gojo xuất hiện lại cho nên cảm hứng để viết phần tiếp theo của chuỗi bài tập lớn này cũng cạn dần. Những thuật thức như *thập chủng đồ thị*, *xích huyết ma trận*,... vì vậy cũng chìm vào quên lãng. Do đó, chúng ta sẽ quay lại với cách xử lý mạch truyện truyền thống khi không biết viết gì tiếp theo chính là **hồi tưởng lại quá khứ**.

Hình 1: Gojo and Sukuna.[1]

The story will revolve around Gojo and Sukuna when they initially opened and owned their own restaurant, before deciding to collaborate, as depicted in the content of Assignment 1

3 Introduction

In this assignment, students will still simulate the process of arranging customers in a restaurant. It will be carried out using commands as described in Section 3.1.

The meanings of the abbreviations and data types of the parameters are described as follows:

- **NAME**: a continuous string of Alphabet characters (including lowercase and uppercase letters) without spaces, representing the customer's name.
- **NUM**: an integer with a different meaning and range of value for each different command.
- **MAXSIZE**: a positive integer, describing a maximum number of area in each restaurant.

Note: after processing all the commands in the input file and outputting the results to the screen, the program must destroy all dynamically allocated data objects, ensuring no garbage in memory before ending program.

3.1 List of command

LAPSE <NAME>:
This command is used to determine the restaurant that the customer will choose based on the customer's name. The process for executing this command is as follows

1. Standardize customer names:

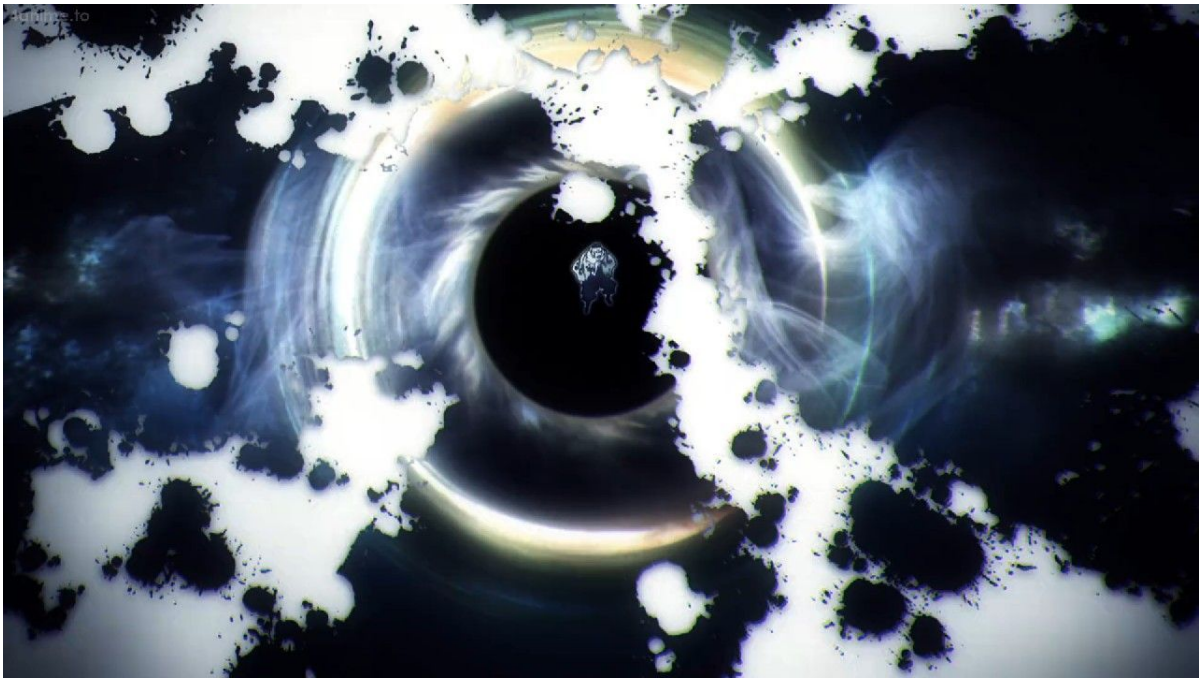
1. First, the employee will count the frequency of each individual character in the customer's name (considering case sensitivity). Then, these characters will be rearranged in ascending order. In case of multiple characters having the same frequency, the character that appears first will be positioned at the forefront of the others.

   For example: Customer's name is **"abaaabbbDd"**, the result in this step will be:
   a : 4 $\longrightarrow$ b : 4 $\longrightarrow$ D : 1 $\longrightarrow$ d : 1.

2. Furthermore, to enhance the security of customer information, employee will apply Caesar cipher to encrypt each character in the created list. The number of shifts for each character is equal to its number of occurrences. Let's denote the result after encoding as list X."
3. Based on list X, start building the Huffman tree as described in the document [4], chapter 5.6. However, there will be certain changes as follows:

- During the process of constructing the Huffman tree, when there are multiple elements with the same value, the element that appears first will be prioritized in the calculations.
- Moreover, the employee will always rotate the tree (including the subtree when constructing the final tree) if it is imbalanced. The mechanism for detecting imbalance and performing tree rotation is precisely like AVL. Let's denote the result as Tree X.

4. Based on Tree X, convert each character in the customer's name (already encrypted) following the outlined in the documentation [4], chapter 5.6.2, result in a binary string. Take up to 10 character from that result, starting from the right and convert the result to decimal value. Refer to the obtained result as Result.
5. Note, the customer's name must contain at least 3 or more different characters. Staff will refuse to accept customers if the customer's name does not meet the requirements.



Hình 2: View in Gojo's restaurant.[2]

2. Choose the restaurant:

Each restaurant will have different decorations, while Gojo's restaurant (Unlimited Void - Res G) aims for a luxurious, elegant style, Sukuna's restaurant (Malevolent Shrine - Res S) aims for a dark, ghostly style.

- If the Result is an odd value, then the customer will prefer to go to res G, otherwise, the customer will go to res S. Each restaurant have a MAXSIZE area.

- Because both restaurant owners are proficient in using domain expansion, they do not limit the number of customers they can serve in each area.

Hình 3: View in Sukuna's restaurant.[1]

3. Choose the area:

After choosing a restaurant, customers will be assigned to specific areas by staff. The area number (ID) will be calculated according to the formula **ID = Result % MAXSIZE + 1** with ID starting from 1 to MAXSIZE. And each restaurant will have a different arrangements for its customers, specifically as follows:

For **Unlimited Void** restaurant:

- Gojo chooses to use a hash table, each value will represent for each area.

- When multiple guests are assigned to the same area, Gojo employs the Binary Search Tree (BST) with the Result value of each guest serving as the key value for constructing the BST. In case a later node has a value equal to an existing one, it is added to the right side. The node with the smallest value in the right subtree will be used to replace when executing node deletion in the BST.

For **Malevolent Shrine** restaurant:

- Sukuna use a min-heap to organize guest areas, where each area is represented as a node in the min-heap and is labeled from '1' to 'MAXSIZE.' When a guess entered or removed from specific area, the value of that node is adjusted by 1 unit accordingly. Only areas with guests are included in the min-heap.
- For example, assume that there are 6 areas (from 1 to 6) in res S, when 3 guests enter area 1, the node labeled '1' with a value of 3 is added to the min-heap. Then, if 2 guests enter area 2, the node labeled '2' is added to the min-heap and go to the root due to re-heap mechanism.
- When changing in the number of guests in an area, the staff will perform a re-heap. Let NUM denote the number of times a certain area receives/removes customers. During a re-heap up (moving an element from a leaf node to the root node), the child swaps positions with the parent when the child's NUM value is smaller than its parent. During a re-heap down (moving an element down to a leaf node), the parent swaps with its child when the parent's NUM value is larger. **In cases where the parent has two children, the parent swaps with the child, which has the larger value. If nodes have the same NUM value, the larger element is the element added to the heap earlier**. The maximum size of the min-heap is MAXSIZE.
- Furthermore, when a certain area is added to the min-heap but after a period of time, no more guests are stayed in that area, the staff will remove that area from the min-heap.

KOKUSEN:
This command is used for G's staff to remove certain guests that Gojo thought were spies from S's restaurant to disrupt his business.

The procedure for executing this command is as follows:

- Initially, convert the BST into an array of values in post-order. Then, compute the number of permutations of the generated array. These permutations, when added sequentially in index order from smallest to largest, should reconstruct the same BST tree as the original.

Hình 4: Angry Gojo.[3]

Example: if the original BST is (2 (1 3)), with "2" as the root, "1" as the left child, and "3" as the right child, and the converted array has the values (1, 3, 2), the staff identifies 2 permutations that can reconstruct the same BST tree as the original: (2, 1, 3) and (2, 3, 1). The array (1, 2, 3) would yield a different BST tree and is thus excluded from consideration. In this case, the returned result is 2, denoted as Y.

- Subsequent to the above, the employee proceeds to remove Y customers from that area in FIFO order. If Y exceeds the total number of nodes in the tree, the entire tree is deleted.
- The above process is iterated for all remaining guest areas in res G.

KEITEIKEN <NUM>:

This command is used for S's staff to remove NUM customers from NUM areas that SUkuna

thought were spies from S's restaurant to disrupt his business. The value of NUM ranges from 1 to MAXSIZE.

Hình 5: Angry Sukuna.[1]

The procedure for executing this command is as follows:

- Staff will select the area that has been unused for the longest time and has the fewest guests to remove guests. For example, assume that res S has 3 areas, with 3 guests enter area 1 , 5 guests enter area 2, and 3 guests enter area 3. This is also the order of arriving guests in the 3 areas
- When the staff receives the command **KEITEIKEN 1**, they will search and find that the area with the minimum number of guests is area 1 and area 3. However, since area 3 has recently had a customer (just used), the staff will select area 1 and remove 1 guest from this area (if there are multiple guests to remove, the staff will remove them in FIFO order).
- If NUM is greater than the number of guests in the area, the staff will consider remove all guests from that area.
- When completely executing this command, print the information of the first to last removed guests in the format: "Result-ID/n" with Result value is described in the **LAPSE** command, and ID is the label of the area.

HAND:

Print the values in the Huffman tree in order from top to bottom, left to right in the format: "char-freq/n" of the customer who most recently visited the restaurant (including restaurant G and restaurant S). Where *char* is the character (before encoding) and freq is the number of times that character appears in the customer's name.

LIMITLESS <NUM>:

Print the BST at area NUM of res G in in-order with the format 'Result/n' at each node. If NUM does not exist in res G or in an area where NUM has no visitors, nothing needs to be done.

CLEAVE <NUM>:

Using the preorder traversal technique, then print information of NUM customers (in LIFO order) in each area in the min-heap with the format: "ID-Result/n" whereID is the label of the area, and Result value is described in the **LAPSE** command. If NUM is greater than the number of existing guests, all guests in the area will be printed.

4 Conclusion

Hình 6: Have a break, Have a Gojo.[2]

4.1 Requirements

To finish the assignment, students need to:

- Download and unzip the initial.zip.
- There are 4 files: main.cpp, main.h, restaurant.cpp and test.txt.. You MUST NOT modify files main.cpp and main.h.
- Modify files Cache.h and Cahe.cpp to implement the cache memory.
- Make sure that there is only one **include** directive in restaurant.cpp that is **#include "main.h"**. No more include directive is allowed in this file.
- The environment used for grading is g++ (MinGW-W64 i686-ucrt-posix-dwarf) 11.2.0.

5 Submission

Students submit only 1 file: **restaurant.cpp**, before the deadline specified in the link "Assignment 2 Submission". There is simple test case used to check your solution to make sure your solution can be compiled and run. You can submit as many as you like but just the last submission is marked. As the system cannot response too many submissions in the same time, you should submit as soon as possible. You will take your own risk if you submit on the time of deadline. After the deadline, you cannot submit anymore.

6 Harmony

There is a question in the exam that are related to the content of the assignment. It will be clearly stated that the question is used to harmonize the assignment. The scores of harmony questions will be scaled to 10 and will be used to recalculate scores for the assignments. Specifically:

- Let x be the score of assignment.
- Let y be the score of harmony question after scaling to 10.

The final assignment score will be:

$$\mathbf{Assignment\_Score = 2 * x * y / (x + y)}$$

7 Other regulations

- Students must complete this assignment on their own and must prevent others from stealing their results. Otherwise, the student treat as cheating according to the regulations for cheating.
- Any decision made by the teachers in charge of this assignment is the final decision.
- Students are not provided with test cases after grading, students will be provided with the assignment's score distribution.

THERE ARE NO EXPLANATIONS OR EXCEPTIONS.

References

[1] https://www.deviantart.com/

[2] https://www.pxfuel.com/

[3] https://www.wallpaperflare.com/

[4] Data Structures & Algorithm Analysis by Clifford A. Shaffer - Edition 3.2 (C++ Version)

——————————END——————————