DOCUMENTATION DS PROJECT

Muhammad Ismail Ramzan

Talha Pasha

20I-0941 – 19I-2034 Cysec T

Contents

Documentation

This is the documentation of the Project of Data structure assignment. Code is fully commented. It is not necessary to read this documentation. Code can be easily understood 😊.

Basic Features

- ➔ Circular singly linked list is implemented for machine
- ➔ Circular linked list contains a doubly for Routing and an AVL tree for data
- ➔ File system is implemented that saves the data
- ➔ User can enter the key value pair and save the data
- ➔ User can manually assign id's & automatically
- ➔ Routing table is implemented to fast the storage & search algorithm's
- ➔ AVL trees are used to save the data of the user
- ➔ Add & Remove new Machine can be done & user can see routing & AVL tree any time of any machine

Working of the code

This documentation will explain the working of the code. From here we will look into the code and will document that how we did the implementation using different data structures.

Circular Linked List:

At first, I have implemented the circular linked list. This is used to represent the machines that are in the system. Each Machine is represented as the node of the circular linked list. Each Linked list 3 Important things

- ➔ Doubly linked list Object
- ➔ AVL TREE Object
- ➔ AVL root pointer for each Machine

Now let’s take a look at the code to understand the structure better.

```
struct circular_linked_list_node
{
    // avl tree fro storing the data...
    AVL avl_tree;
    AVL_node *avl_root = NULL;
    // A routing_Table_in_each_machine
    Routing_table_linked_list route_table;
    int machine_id;
    // a pointer to point on the next Machine
    circular_linked_list_node *next;
};
```

Machine id is just the id of the machine so we can differentiate between different machines.

Functions of Circular Linked List

How did we take machine id manually?

```
 // A function that Makes sure that the data is entered in ascending Order always
    void insert_ascending_order(int new_machine_id)
    {
```

This is the insert function which worth discussing and mentioning. This function takes in the machine Id from the User and then find the correct space where the machine node to be inserted. Just like we do the operations in singly linked list to arrange them. This was similar to that.

User may enter Repeated or OUT OF RANGE id's so to tackle this problem we added some important check's. This loop is used to check weather the data entered by the user is out of order range or not. It also check's weather the id has been assigned to user or not.

```
 while (true)
            {
                // cheaking weather the id is repeated or not!!!
                bool is_machine_already_present =
Machines.check_if_machine_present(machine_id);
                // if machine is already present..
                if (is_machine_already_present)
                {
                    cout << "\n Machine id is already in the list.. \n Enter
Again:";
                    cin >> machine_id;
                }
                else if (machine_id <= -1 || machine_id >= (max_range_of_machine
- 1))
                {
                    cout << "\n You have Entered Id which is not In
Range..\nEnter Again: ";
                    cin >> machine_id;
                }
                else
                {
                    break;
                }
```

Taking the Id Automatically

This one was pretty easy. We used the rand function to create the machine within the range and just like the above machine we created the check's so that the id is not assigned again or are not repeated.

```
            // 2 Checks's
            int random = rand() % (max_range_of_machine - 1);
            //  Must be In range..
            // must not be repeating..
```

The main menu Implementation

This is the main menu. It uses the simply switch case in C++ that helps us to run the different functions or cases at different user input.

```
        cout << "\n Please specify the Operation that You want to Perform \n";
        cout << "1) Add A Machine into the System [o]";
        cout << "\n2) Remove A Machine from the System [o]";
        cout << "\n3) Insert a Key,Value Pair into Machine [o]";
        cout << "\n4) Search for Key Into the AVL Tree (Machines) [o]";
        cout << "\n5) Delete the Key from the Machine [o] ";
        cout << "\n6) Print Routing Table of any machine Id [o] ";
        cout << "\n7) Print the Avl  tree of the Machine [o]";
        cout << "\n8) See machines id;s ";
        cout << "\n9) Exit\nEnter:";
```

The main reason of showing this menu that Now will discuss each feature one by one. So, we can better understand the code & the working.

Working of Routing Table

There is a function in our Program which is responsible for the working og the routing table.

```
 void fill_routing_table(int no_of_machines, int max_range_of_machine, int
bit_Size)
```

This function calculate the routing index and Routing values and then assign them to particular machines.

Note: Routing table class has a pointer of machine type that points towards the machine that is on the index.

Adding a Machine into the system

First of all , we put the checks that the user do not enter machine out of range or the id should not be repeated Then after that we used the function

```
Machines.insert_ascending_order(mac_id);
```

To insert the machine into the system. Then we displayed the machine. Is that's all? Off Couse not! We needed to fix the Routing Table and AVL Machine data.

For routing table, we used a function that calculates the Routing value for each machine and then Update the Table of each machine. It performs special calculation for the root.

After the machine has been added we updated the routing table and for avl we deleted all the AVL nodes then used the file system to get the values back into the AVL tree of corresponding.

Removing a machine from the system

Removing a machine from the system is similar to adding. It's easier than the addition though. We just deleted the machine and store it's AVL value into a file system. Then we iterated the file system and assigned the data id to the machines which are responsible for handling them.

```
   Get the deleted tree nodes pointer
                    tmp->avl_tree.deleteTree(tmp->avl_root);
                    Machines.del(tmp);
                    // First of all i need to delete the Previous One..
                    Machines.delete_routes(bit_size);
                    cout << "\n Updating Routing Tables \n";
                    Machines.fill_routing_table(number_of_machine,
(max_range_of_machine - 1), bit_size);
                    Machines.assign_avl_values_of_machine_to_other(tmp,
bit_size);
                    // Now obj has the tree... We need to get the data and insert
it into the machine

                    // we have the data into file as well.. Let's read from there
```

Storing the data into the Machine

For storing the data. We first iterated the routing table and moved according to the description given into the project system. Once we find that particular machine then we insert the value into the AVL tree of the respected machine that we found after the calculation.

```
  temp = Machines.find_storage_machine(data_id, start_machine_id, bit_size);
            // Now we can store that data into the temp Node..

            temp->avl_root = temp->avl_tree.insert(temp->avl_root, data_id,
data);

            // creating the file
            Machines.create_temp_file(temp->machine_id);
            temp->avl_tree.show_in_file(temp->avl_root, temp->machine_id);
            temp->avl_tree.show(temp->avl_root, 1);
```

After storing the data into the AVL tree we also store the data into the file system for further use & the requirement of the project.

Searching the data from the machine

For searching the data, I calculated that which machine will be responsible for storing them. So, using the routing table there was the search. Once we read at the machine, we get the machine address. In that particular machine we iterate the AVL tree and hopefully find the data if it exists in the system.

```
 cout << "\n Search the Value from the System\n";
            // Now i need to Travel through the Routing table and find where the
key is stored on the system..
            cout << "\n Please Enter the Key to Seach for \n";
            int data_id = 0;
            cin >> data_id;
            data_id = hash_function(data_id, max_range_of_machine);
            circular_linked_list_node *temp = NULL;
            temp = Machines.find_storage_machine(data_id,
Machines.get_head_machine_id(), bit_size);
            // Now we will Traverse the Avl tree of them Temp..
            temp->avl_tree.find_a_key(temp->avl_root, data_id);
```

Deleting the data from the machine

By using the above searching algorithm we find the key. Once we find the key, we use the delete function of AVL tree that is responsible for the delete of the data from the system.

```
            int count = 0;
            temp->avl_tree.return_count_of_key(temp->avl_root, data_id, count);
            cout << "\n There are total " << count << " Keys";
            cout << "\n Please enter the value to Delete the Key/value
Permanetly";
            cin.ignore();
            string val;
            getline(cin, val);
            temp->avl_root = temp->avl_tree.deleteNode(temp->avl_root, data_id);
```

In the above code snippet delete Node is responsible for the removal of the data.

Displaying the Routing table

First, we check if the machine exists, if it does then we simply call a function. That function iterates across the machine and prints out the doubly linked list inside of it.

```
// this Function display's the Routing table of any machine..
            bool is_exist = Machines.check_if_machine_present(temp_id);
           //
           if (is_exist)
           {
              Machines.display_me_routing_table_of_temp_id(temp_id);
           }
           else
           {
               cout << "\n This Machine does not exist in the system \n";
           }
```

Displaying the AVL tree

Displaying the AVL tree was pretty much easy. At first, we checked weather the machine exists or not. If it does then We call the AVL display function of that particular machine that user has asked for.

```
int temp_id = 0;
           cout << "Please Enter the Id of the Machine for ";
           cin >> temp_id;
           bool is_exist = Machines.check_if_machine_present(temp_id);
           //
           if (is_exist)
           {
               Machines.display_me_avl_machine(temp_id);
           }
           else
           {
               cout << "\n This Machine does not exist in the system \n";
           }
```

Displaying all the Machine

This might be the easiest. I simply added a loop and printed the Circular linked list nodes as they are represented the nodes in the system.

```
Machines.display_machine_nodes();
```

Summary

In this doc we have discussed how we did our implementation. Remember, we have not included the information of Linked list or AVL tree functions because they have been taught us in class & easily available. We discussed the main things that how we joined the pieces and made up the Ring Distributed system which was the main focus & required.

Doing this Project was Really Fun. It was a great Example of Creativity. 😊

Thanks