**EVEN SEMESTER - 2022**

**LOGIN & PLAY**

**MTE PROJECT REPORT**

**Course Name : Algorithm Design and Analysis**

**Course Code : CO - 208**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER ENGINEERING**

**Submitted by**

**Vinayak Sharda - 2K20/CO/489**

**Vivek Kumar Thakur - 2K20/CO/493**

**Under the supervision**

**of**

**Dr. Ashish Girdhar**

****

**DEPARTMENT OF COMPUTER ENGINEERING**

**DELHI TECHNOLOGICAL UNIVERSITY**

**(Formerly Delhi College of Engineering)**

**Bawana road, Delhi-110042**

**CANDIDATE’S DECLARATION**

We, Vinayak Sharda (2K20/CO/489) and Vivek Kumar Thakur (2K20/CO/493), students of B.TECH (COE) declare that the MTE Project Report titled “Login & Play” which is submitted by us to the Department of Computer Science and Engineering, Delhi Technological University, Delhi is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Fellowship, or other similar title or recognition.

**Place: DTU, Delhi                                                        VINAYAK SHARDA -2K20/CO/489**

**Date: 25th April 2022                                VIVEK KUMAR THAKUR – 2K20/CO/493**

**CERTIFICATE**

I, hereby certify that the Project titled “Log in & Play” submitted by Vinayak Sharda (2K20/CO/489) and Vivek Kumar Thakur (2K20/CO/493), to Department of Computer Science and Engineering, Delhi Technological University, Delhi, as part of Innovative Work is a record of project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

**Place: DTU, Delhi                                                                                Dr. Ashish Girdhar**

**Date: 25th April 2022                                  (Department of Computer Engineering)**

**ACKNOWLEDGEMENT**

We are very thankful to Dr. Ashish Girdhar (Department of Computer Engineering) and all the faculty members of the Computer Science Engineering Dept. of DTU. They all provided immense support and guidance for the completion of the project undertaken by us. It is with their supervision that this work came into existence. We would also like to express our gratitude to the university for providing the laboratories, infrastructure, test facilities, and environment which allowed me to work without any obstructions. We would also like to appreciate the support provided by our lab assistants, seniors, and peer group who aided us with all the knowledge they had regarding various topics.

**VINAYAK SHARDA (2K20/CO/489)**

**VIVEK KUMAR THAKUR (2K20/CO/493)**

**INDEX**

1. Introduction
2. Objectives
3. System requirements and software used
4. Implementation and Execution
5. Limitations
6. Conclusion
7. References

**INTRODUCTION**

Our project aims to create a gaming environment that is user friendly and interactive. In order to play the game, the user has to log in with his account. However, if he is playing the game for the first time, then he has to register his account in the registration portal. Once he has successfully logged in, he can play the game. The game is a 2048 game whose objective is to slide numbered tiles on a grid to combine them to create a tile with the number 2048; however, one can continue to play the game after reaching the goal, creating tiles with larger numbers.

**OBJECTIVES**

* To implement a login portal using trie data structure.
* To implement a 2048 game using a 2D matrix.

**SYSTEM REQUIREMENTS AND SOFTWARE USED**

* The program runs on Windows 10 operating system.
* The program should run on 32-bit or 64-bit architecture system, 64-bit
* system was used to create the program.
* At Least 2GB system RAM should be available and the Processor should be faster than 1.6 GHz.
* Visual Studio Code has been used for creating programs in C++.

**IMPLEMENTATION AND EXECUTION**

The project can be broadly divided into two parts:

1. Log in portal
2. 2048 game

Their incorporated execution can be understood by the flowing flowchart.

**Here is the flowchart of the program:**







**Working of Login Portal**

In order to log in, the user should have his username and password registered with the program. While registering, the model stores the username and the password of the user. Trie data structure is used for this functionality and when the user wants to play the game again he can simply access it by entering his username and password.

Trie is an efficient information reTrieval data structure. Using Trie, search complexities can be brought to optimal limits (key length). If we store keys in binary search tree, a well balanced BST will need time proportional to M \* log N, where M is maximum string length and N is the number of keys in the tree. Using Trie, we can search the key in O(M) time. However, the penalty is on Trie storage requirements.

Inserting data into trie:

* 1. Suppose the username is ‘vivek’ and we want to store it. In our program the trie-root has a children array of size 256.
  2. ‘v’ is stored in trie-root -> children[v] (ASCII code of ‘v’ is used as index).
  3. Now ‘v’ is itself a considered as a root and ‘i’ is stored in its children array.
  4. Similarly, ‘v’, ‘e’ and ‘k’ are stored.
  5. In case of ‘k’, bool isEndOfWord is marked as true.
  6. Similarly, passwords are also stored in the trie.

Code snippet:

*void* insert(TrieNode \**root*, string *username*, string *password*)

    {

        TrieNode \*temp;

*int* index = *int*(*username*[0]);

        if (*root*->children[index] == NULL)

        {

            temp = getNode();

*root*->children[index] = temp;

        }

        if (*username*.size() == 1)

        {

*root*->isEndOfWord = true;

            if (!*password*.empty())

            {

                insert(*root*->children[index], *password*, "");

            }

            return;

        }

        insert(*root*->children[index], *username*.substr(1), *password*);

    }

Similarly, the search function can also be implemented.

Time Complexity of insert function is O(m) where m is the length of the string.

**Working of the 2048 game**

After the login is successful, the user is provided with a 6\*6 matrix where the value of one cell is 2 and one another cell is 4, both of these cells are selected randomly from the grid.

Now the user is provided 5 options:

1. To move all the cells up
2. To move all the cells down
3. To move all the cells right
4. To move all the cells left
5. To quit

**What does a successful move do?**

If a move in a particular direction is successful then all the mergeable cells are merged and empty cells are filled by non-empty cells along that direction.

The condition of cells to be mergeable is that both the cells should have the same value and should be adjacent to each other. The value of the first cell of the two along that direction is doubled and the second cell’s value is set to zero i.e., it is now empty.

Here is the code snippet of ‘move\_down’:

void move\_down(int grid[6][6])

{

    int i, j, lr, cc;

    //lr indicates the last cell of the current column

    //cc indicates the current column

    for (j = 0; j < 6; j++)

    {

        lr = 5, cc = j;

        for (i = 4; i >= 0; i--)

        {

            if (grid[i][j] != 0)

            {

                if (grid[i + 1][j] == 0 || grid[i + 1][j] == grid[i][j])

                {

                    if (grid[lr][cc] == grid[i][j])

                    {

                        grid[lr][cc] \*= 2;

                        grid[i][j] = 0;

                    }

                    else

                    {

                        if (grid[lr][cc] == 0)

                        {

                            grid[lr][cc] = grid[i][j];

                            grid[i][j] = 0;

                        }

                        else

                        {

                            grid[--lr][cc] = grid[i][j];

                            grid[i][j] = 0;

                        }

                    }

                }

                else

                    lr--;

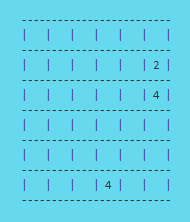
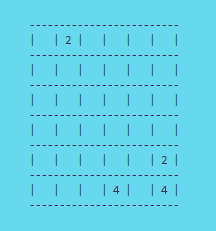
            }

        }

    }

}

Here are is output when the above algorithm is executed:

Before After

If the move is successful (that we check by simply comparing the grid before and after the move) then we insert either 2 or 4 at an empty location.

After that we check whether any further move is permissible by searching for all the empty locations and mergeable pairs of cells. If any of the conditions are met then the user is provided the 5 options again, else the game is over and the user’s score is displayed which is the largest integer value in the grid.

All of the functions that are:

* move\_up,
* move\_down,
* move\_left,
* move\_right,
* compare,
* insert,
* filled,
* highscore,
* view\_grid

are implemented in O(n\*n) time.

**Limitations**

* Once the program is terminated, all the usernames and the passwords are also deleted.
* The user can not modify his username or password
* The user can not undo his last move.
* The game has no objective i.e., even though it is named as ‘2048’, the target is not to reach to number 2048, the user can keep playing on.
* The high scores of the players are not being stored.

**Conclusion**

By implementing this project, our understanding of data structures, especially tries and arrays improved significantly and we also learnt the algorithm of the 2048 game.

**References**

<https://www.geeksforgeeks.org/trie-insert-and-search/>

<https://en.wikipedia.org/wiki/2048_(video_game)>