Data Structures Semester Project

Aiman 21K-3906

Huda 21K-3907

2048 Game in java

Problem Statement for 2048 game

The problem is to design a console based “2048 game” in java implementing different Data structures concepts.

2048 is a game in which the player moves numbered tiles on a 4x4 grid in four possible directions (up, down, left, and right). The goal is to combine the tiles to form a tile with the number 2048. Tiles can move as far as they can in the chosen direction until they are stopped by another tile or the grid's edge. If a tile is stopped by another tile of the same value, the two tiles will merge into one tile with the sum of their values.

Objectives of the Game

FOLLOWING ARE THE OBJECTIVES OF GAMES:

- Create a grid of 4x4 to play the game on the grid.
- Generate the random tiles of 2s and 4s on the grid.
- Merge the same numbered tile to create a new high tile.
- Shift the tiles to the left when player play left move.
- Shift the tiles to the right when player play right move.
- Shift the tiles up when player play up move.
- Shift the tiles to down when player play down move.
- Only merge two Adjacent tiles on the grid.
- Store the player scores in the stack and keep updating the score.
- Let player change the gaming keys through setting.
- Let the player undo the move.

Data Structures concepts used in the game

LinkedList:

Singly-Linked lists are used in a variety of functions. We are using the linked list to get the directions/locations of the grid's empty rows and columns so that we can place the next random tile on the grid. The linked list provides us with valid adjacent locations. The linked list is also used when shuffling the tiles. The linked list provides empty locations for the tiles to shuffle/move to.

Stacks:

Stacks are used to keep track of the Score and to update the player's scores. If a player reverses the move, the last entered score in the Stack will be the first score to pop out. The top function of the stack also allows the player to see the score on the board.

Stacks are similarly used to keep track of the tiles on the grid. If a player reverses the move, the last tile to enter the board will be the first to pop out. Singly linked lists are used to implement stacks.

Hash Maps:

The gaming keys are stored using Hash Maps. Players can use the setting function to change the gaming keys 'A', 'W', 'S', and 'D'. To play the game, the player can change these gaming keys to any other keys of characters or integers. The keys will be stored within the hash map. Only for that specific game. The function will prompt the player to enter the keys they wish to use, and the hash map will store the keys. Hash maps insertion and retrieval take O (1) time complexity and space complexity is O (n).

Recursion:

In-direct recursion call is used in different functions. To merge the tile of same numbers again and again. To generate the random tiles of 2s and 4s recursion call is being made. While generating the board again and again in-direct recursion call is being made.

Time and space complexity of the project:

Singly-Linked list space and time complexity:

Each node in a linked list contains two pieces of information (the value and the pointer). This means that the amount of data stored grows in a linear fashion as the number of nodes in the list grows. As a result, the linked list's space complexity is linear: O(n) is the dimension of space. In a singly linked list, the time complexity for inserting an element from the list is O(n).

Stacks space and time complexity:

Operations like insertion or deletion in a stack take constant time i.e., O (1). Space complexity for each operation in a stack is O (1), as no extra space is required for any operation.

Hash maps space and time complexity:

HashMap, we can achieve an average time complexity of *O (1)* for the *put* and *get* operations and space complexity of *O(n)*. Hash set is another good substitute as it does not allow duplicate values, but the space complexity would be O(n) and since Hash set has space complexity of O(n) and Hash maps are much faster as compare to hash set. The reason

that HashMap is faster than HashSet is that the HashMap uses the unique keys to access the values. It stores each value with a corresponding key and we can retrieve these values faster using keys during iteration. While HashSet is completely based on objects and therefore retrieval of values is slower.

Other 2048 Games in the market:

There are many other application of 2048 games available on play store implementing different technologies, and it can also be played on the browser as it is a very popular game. Our goal was to create 2048 game in java implementing various different types of data structure. We made a console-based java application of 2048 game. The time and space complexity of this game depends on the recursion call that are being made at the time of playing it, also the time and space complexities which are being used by the other data structure are mentioned above.