Assignment 4

Subhalingam D

September 30, 2019

1 Tries

1.1 `Person`

Contains the name and phone number of a person. The `getName()` returns the name of the person and `toString()` is overridden to return a string of required format.

1.2 `TrieNode`

The children are stored in `ArrayList` of size 95. It also stored the value at the leaf. It has some trivial methods and a constructor:

- `TrieNode()`: Initialises the value and all children to null
- `setValue()`: Sets the value
- `getChild()`: Sets the child of the given character
- `makeChild()`: Make a child of the given character
- `hasChild()`: Returns if it has a child (when there is a child that is not null)
- `hasChildOtherThan()`: Checks if there is a child other than given child (index)
- `killChild()`: Sets the child (index) to null
- `removeValue()`: Removes the value
- `getChildByIndex()`: Gets the child by given index (number)
- `getValue()`: Gets the value

Note: When character is bassed, it's index is determined by a simple mapping: $h(i) = i - 32$..

1.3 `Trie`

The `root` is the beginning of a `Trie`.

1.3.1 `insert()`

Maintain a `curr` node initialised to `root`. For $0 \leq i < size$ (where $size$ is the length of the string to be added), if `curr` does not have the child for character at $(i+1)^{th}$ position in the word, make one. Then make `curr` point to that child. After $size$ iterations, the loop is terminated and `curr` has reached the end for this word. Now, if there is a value already at this place, return false; else set the value and return true.

1.3.2 `delete()`

Start from `root`. For $0 \leq i < size$ ($size$ being the length of the string), if there isn't a corresponding child, then the given word doesn't exist. If there is a value or other children while travelling down, make a note of this (we will be needing the last such occurrence). After the loop gets terminated, check if there is some `value` at this node-if not there, return not found. If there is/are further child(ren) for this node, just remove the `value` present; else remove the corresponding child from the node whose occurrence we had noted above.

1.3.3 `search()`

Starting from `root`, travel down the `Trie` for length of the search string. While travelling, if a corresponding child is not present, it means that our search string isn't present. Return accordingly. If after the end of travel, if there is no `value` at this Node, then the search string is still not there. If everything has been passed, then return the `value`.

1.3.4 `startsWith()`

Similar to `search()` travel down and return the matching node or `null`. Then `printTrie()` is called.

1.3.5 `printTrie()`

Do a recursive on this function with the Node `trieNode`, as follows:

- if `trieNode` is `NULL`, return
- if `trieNode.getValue()` is `NOT NULL`, print this `value`.
- Call `printTrie()` on each of the children.

It is a basic Depth-First Search.

1.3.6 `print()`

Maintain two `ArrayList` of `TrieNode`, say t_1 and t_2 and another `ArrayList`, say c of characters. Add `root` to $t2$ and have a loop until t_2 becomes empty. Print "Level x" on console ($x = 1$ at the beginning and incremented at each loop. Make $t_! = t_2$ and t_2 to new object. For $0 \leq j < 95, 0 \leq it_2.size()$: Get $(j+1)^{th}$ child of $(i+1)^{th}$ index of t_1. If `NOT NULL`, add it in t_2 and add the ASCII value of $j + 32$ to c (if $j \neq 0$). After the loop gets terminated, print all the elements in c by seperating them with commas. Reset c and print a new line.

1.3.7 `printLevel()`

Do a similar thing as `print()` but add `x` $\leq l($`l`$isthe$`l`th `level we need) along with` $t_2.size() > 0$`, in the loop condition. Also print only the level required, rather than printing all the levels.`

2 Red-Black Tree

2.1

3 Priority Queue

3.1 `Student`

`Test class that stores name and marks of the Student.`

- `compareTo() compares the marks of the student with the marks of the student object that is being passed and returns negative value if the student's marks is less than that of the passed object, 0 if equal or a positive value if greater`
- `toString() is overridden to print output in desired format.`
- `getName() returns the name of the student.`

3.2 `MaxHeap`