BAHRIA UNIVERSITY (KARACHI CAMPUS)

|  |  |
| --- | --- |
| **CSC 221 - Data Structures & Algorithms - Assignment 3** | |
| CLO-4 | Deadline: 1st February , 22 |
| Class: BSE-3A/B | Total Marks 10 |

1. A database contains information about all trains leaving the Washington Union station on Monday. Each train is assigned a departure time, a destination, and a unique 8-digit train ID number. Examine which data structures you can use to solve each of the following scenarios.
   1. The schedule contains 200 trains with 52 destinations.
   2. Trains with the same destination should further be sorted by departure time.

Depending on scenario, you may need to either (a) use multiple data structures or (b) modify the implementation of some data structure. Justify your choice.

1. You are playing a game. The game consists of 10 questions which helps to guess the name of famous people. The rules of this game are:

* Your opponent will think of the name of a famous person
* You ask a question with Yes or No
* Your opponent gives you the answer to that question
* Based on your opponent answer, you will ask another yes-or-no question, so on
* You cannot ask more than 10 yes-or-no questions
* If you guess the name of the person, you win; otherwise, you lose.

Compare which data structure can help you to implement the above strategy.

**Solution 1**

a) A dictionary with the keys being the destination and the value being a list of corresponding chains would be one solution. Any implementation of a dictionary would suffice.

We may also modify a separate chaining hash table to have a capacity of exactly 52 and not perform resizing. We could then fill the table and simply print out the contents of each bucket by ensuring that each destination translates to a unique integer from 0 to 51 and hashing each train based on this number.

A third option is to utilize a BST or Red-Black tree, in which each train is compared first on its destination, then on its other characteristics. After we've inserted the trains into a tree, we can use an in-order traversal to print out the trains ordered by destination.

b) Regardless of which solution is modified, the train objects must be compared first by destination and then by departure time.

We can improve our original solution by having the dictionary store the value as a sorted set rather than a list. (A BST or a Red-Black tree would be used to implement the sorted set.)

Using a BST or a Red-Black tree as the bucket type, we may adapt our second approach in a similar way.

Our third approach requires no changes: if the trains are now compared first by destination and then by departure time, the iterator of the Red-Black and BST trees will print out the trains in the desired sequence.

**Solution 2**

**STRATEGY**

The data structure used to implement the above guessing game is a BINARY TREE, in which each root has two nodes.

**EXPLANATION**

• Each question will be held by one of ten roots.

• Because binary trees only contain two child nodes, each node will have two nodes with the values YES and NO.

• If the answer to a specific question is YES, the process will be stopped.

• However, if the last question is again a NO, the game will terminate because there is no more node in the tree.

Diagram

Description automatically generated

1