CSL 214 : Data Structures and Program Design II
Lab Assignment-2

1. Announcement Date: **29-03-2020**

2. Due date: Submit online by **12.00 noon on Monday 13th April, 2020**

3. The assignment has to be done individually.

4. No Copying / sharing of code is allowed for this assignment. If any such case is identified, the original author and person who has copied both will be penalised equally and zero marks will be awarded.

5. You need to submit your source files by attaching them to a mail and sending it on **dspd.assignment@gmail.com** by the common deadline. Though viva will be at different times for different students, the submission deadline is same for all. Please attach .txt, .c and .h files only. No .obj or.exe files should be attached. The subject should be marked as DSPD-2-Assignment-2: Your enrolment no.

6. Dates for assignment viva will be announced later.

<u>Problem for R1 Batch:</u>

A call routing problem

Description of call routing system:

Every mobile phone is connected to the nearest *base station/cell-phone tower*.At any point of time, a mobile phone is connected to only one base station. Whenever a mobile phone moves from the area of one base station to another base station, it will be disconnected from its original base station and connected to the new base station which is nearest to its live location.

When a phone call is made from phone $p_1$connected with base station b_1 to a phone p_2, the first thing that the base station $b_1$has to do is to find the base station with which p_2 is connected. For that, b_1 sends a query to a central server *C* which maintains a data structure that can answer the query and return the name of the base station, let's call it b_2, with which p_2 is connected. *C*is the root node of this hierarchical structure that sends some routing information to b_1 so that $b_1$can initiate a call with b_2 and, through the base stations p_1 and p_2 can talk.

You have to maintain a hierarchical data structure in this assignment. At the lowest level (level 0/leaf nodes) is the individual base station which defines an area around it such that all phones in that area are connected with it, e.g., all the phones that are currently in the VNIT campus are connected with the base station in VNIT. This base station also connects other phones in nearby area. We assume that base stations are grouped into geographical locations served by a level 1 area exchange. So, for example, the VNIT base station (level 0/leaf node) may be connected by the Ambazari level 1 area exchange.Each level *i* area exchange is connected by a level *i*+1 area exchange which connects a number of level *i* area exchanges. e.g., the VNIT level 1 area exchange and the Subhash nagar level 1 area exchange may be connected by a Ambazari level 2 area exchange.A base station can be considered to be a level 0 area exchange in this hierarchical structure.

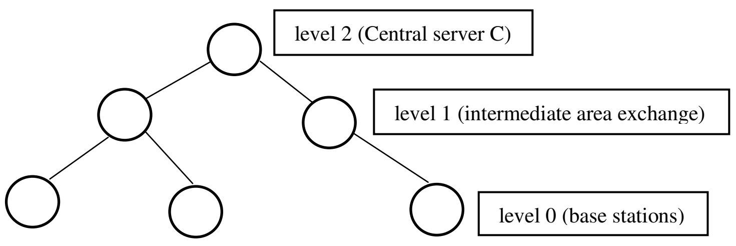
level 2 (Central server C)
level 1 (intermediate area exchange)
level 0 (base stations)

Fig. Example of routing tree structure

Every i^{th} level area exchange, *e*, maintains a set of mobile phones, S_e, as follows. The set S_e is called the resident set of *e*. The level 0 area exchanges (base stations) maintain the set of mobile phones connected directly with them. A level *i*+1 area exchange *e*, maintains the set S_e defined as the union of the sets of mobile phones maintained by all the level *i* area exchanges it serves. Clearly, the root of the routing map maintains the set of all currently connected mobile phones.The process of call routing goes as follows.

I. When a base station *b* receives a call for a mobile phone with number *m* it sends this query to *C(root node).*

II. If the root of the tree is C, we first check if $m \epsilon S_C$. If not, then we tell *b* that the number *m* is "not reachable."

III. If $m \epsilon S_C$ we find the area exchange*e* such that parent(*e*) = *C* and $m \epsilon S_C$, i.e. we find the child of *root node C* which contains *m* in its resident set.

IV. Continue like this till we reach all the way down to a leaf of the routing tree where mobile number *m* resides. This leaf is a base station *b'*. The central server sends *b'* to *b* along with the shortest path in the routing tree from *b* to *b'*.

Implement a data structure that facilitates operations mentioned above using trees (you can choose any tree data structure covered in the class Binary/AVL/B-tree/B+tree or its modification). Your implementation should have at least following functions:

You can assume that the root of the tree is available wherever required.

- *IsEmpty ( )*: returns true if the set is empty.
- *IsMember (MobilePhone m)*: Returns true if *m* is in the set, false otherwise.
- *Insert (MobilePhone m)*: Inserts *m* into the set/tree. (base station can also be i/p).
- *Delete (MobilePhone m)*: Deletes *m* from the set/tree, reports error if *m* is not in the set.
- *MobilePhoneSet residentSet (Exchange e)*: This returns the resident set of mobile phones of the exchange *e*.
- *Exchange findPhone(MobilePhone m)*: Given a mobile phone *m* it returns the level 0 base station with which it is connected or throws an exception if the phone is not found.
- *Exchange lowestRouter (base a, base b)*: Given two level 0 base stations*a* and *b* this method returns the level *i*area exchange node with the smallest possible value of *i*(level-wise) which contains both *a* and *b* in its subtree. If *a* = *b* then the answer is *a* itself. In-short, in tree terminology, this function finds out the common ancestor in the tree.
- *ExchangeList routeCall(MobilePhone a, MobilePhone b)*: This method helps initiate a call from phone *a* to phone *b*. It returns a list of exchanges. This list starts from the base station where *a* is connected and ends at the base station where *b* is connected and represents the shortest route in the call routing tree between the two base stations. It goes up from the initiating base station all the way to the *lowestRouter( )* connecting the initiating base station to the finalbase station and then down again. The method throws an error as appropriate. In tree-terminology, this is like finding shorted path from one node to another.
- *movePhone (MobilePhone a, base b)*: This method modifies the routing tree by changing the location of mobile phone *a* from its current base-station to the base station *b*. As an effect, lot of other data (sets of mobiles maintained at different levels) that is maintained could also change.

<u>Problem for R2 Batch:</u>

Car Showroom Management System:

Stock details in individual showroom:

Inventory of a car showroom includes the following.

1. Details for each car like "Name of car, Colour of car, Price of car, Fuel type of car and Type of car (Hatchback, Sedan or SUV)".
2. Number of sold cars.
3. Number of available cars.

Assume, Inventory of the cars is maintained in a tree. Maintain the VIN (Vehicle Identification Number) as a primary key for each car.

Count of cars should increment and decrement in the database of sold cars and available cars respectively as per the car sale.

Sales person in individual showroom:

Sales person can login and search the details of the stock and customer. Sales person will maintain the record of sold car and its respective customer. Details of sales person maintained are :

1. Id and Name.
2. Sales person sales target. (50 lakhs rupees/Month)
3. Sales person sales achieved. (Rupees in lakhs)
4. Sales person commission is 2% of sales achieved.

Note: Sales person will access data from stock details and customer details tree. Maintain separate tree for sales persons.

Customer details in individual showroom:

Each sales person will keep record of details for his/her customers like Name of customer, Mobile No. of customer, Address of customer, Sold car (VIN) and Car registration number in a tree.

The sales person will also keep the record of sold cars like Car (VIN) and Payment type (Cash/Loan. If car is purchased on loan then following is the criteria.

1. Down payment should be greater than 20% of car price.
2. 9.00% rate of interest for 84 months EMI.
3. 8.75% rate of interest for 60 months EMI.
4. 8.50% rate of interest for 36 months EMI.

Maintain the loan data for each car.

Note: You may decide to maintain separate tree for customers. Each customer may have a customer id.

A. Assume there are three showrooms each selling all types of cars. Create databases for them.
B. Write a function to add new sales person in the sales person database of a given showroom.
C. Find the most popular car among all three showrooms.
D. Find the most successful sales person according to sales. Award him/her with 1% extra incentives as per sales achieved and print the same.
E. Write a function sell() with input details of sales person and customer.
F. Based on sales figures of previous months, write a function to predict the next month sales for each showroom.
G. Write a function with one argument (VIN) which will display all information related to the car, whether it is in stock or sold, customer name, sales person name, loan type etc.
H. Write a range search function that takes two VIN numbers and print the information for cars having their VIN numbers in the given range.

Note: You can decide to use any of the AVL, B or B+ trees to maintain all databases.

<u>Problem for R3 Batch:</u>

Parking Lot Design

An apartment has 3 wings (wing-A, wing-B, wing C) with 10 floor each (9 residential floors and ground floor is dedicated for parking vehicles). On each floor there are 4 flats (building capacity: 3*9*4=108).It has three parking levels such that Level A dedicated for wing –A , Level B for wing –B and so on. Each parking level has multiple compact and large spots.

Parking Condition: Two-wheelers can be parked in any spots. Four-wheelers can be parked in one large spot or two consecutive compact spots.

Details need to be stored:

- **Parking_Lot details** to store parking spots information like Parking_level, dedicated_wing, no_of_compact_spots, no_of_large_spots, status_compactspots(array storing which spots are occupied or not), status_large_spots(array storing which spots are occupied or not).
- **Flat_Details** to store floor and wings details like wing_id, flat_id, no_of_two_wheelers, no_of_four_wheelers.

Write a program to implement following functions-

1. **Add_or_map_vehicle_Node()** : whenever new vehicle is purchased by flat owner it should be added to corresponding parking spot as per parking condition given above. If any flat occupant cannot get parking spots in their respective parking level then remaining parking levels in order (i.e. if A is full then first B is searched and then C) should be taken into consideration. **If any occupant has more than 1 four wheeler then only one large spot should be allocated in parking level dedicated to wing and others should be allocated other than dedicated parking level**. This function will map permanent parking spot to flat owner vehicles but their status will not be allocated all time. At the end this function should display vehicle mapping status for all levels in tree representation.
2. Write a function **Park_vehicle ()** which will allocate mapped parking spot to vehicle just by reading vehicle number.
   Note:

- If the vehicle number is not mapped with any parking spot then it is considered as guest vehicle. In such cases any free spot depending on vehicle type will be allocated.
- For mapped vehicle if the spot is not free, then first free spot in any parking level can be allocated (but the search should start with vehicle owners dedicated parking level).
- This function should keep track on date, time, insiders and visitors, vehicle number for visitors.

3. **Remove_vehicle_node ()**: if any flat owner sells his vehicle then corresponding entry from parking tree should be removed.
4. Write a function **Display_Visitors()** to get all visitors on particular date along with their vehicle type and number and parking level , parking spot details.
5. Write a function **Parking_status ()** to display the list of free and allocated parking spots in each parking level.

Note : You can decide to use any type of trees to store parking and/or flat owner database.

<u>Problem for R4 Batch:</u>

TOKEN BASED AUTHENTICATION

An "XYZ" organization provides services to its customers by using token based authentication process. There are **four membership levels** of customers **PLATINUM, DIAMOND, GOLD** and **REGULAR**. Each membership level has **two sub access levels** e.g. L1 and L2. To avail any service user has to generate a token by supplying USER_ID and PASSWORD. If any new token is generated it should be added to its corresponding membership level in tree representation. At any instance maximum **'n'** customers are allowed to avail services in each membership sub-access level, they can be called as **active members** and others will be added to **waiting customers**. For each user following information needs to be maintained-

- User_ID
- Password
- Membership level
- Sub access level
- Date_of_membership
- Renewal date

For token database-

- Token no.
- Arrival time
- Waiting time

Write a program to implement following functions-

1. **add_customer_to_tree ():** whenever token is assigned to customer, it should be added to proper membership level in a tree.
2. **display_active_and_waiting_customers ():** to display list of active and waiting customers in membership level
3. **move_to_membership ():** If the renewal period of membership for PLATINUM, DIAMOND and GOLD user is over then the particular user should be moved to REGULAR category in any sub access level.
4. **Update_membership** () DIAMOND and GOLD members can change their membership anytime. GOLD members cannot directly avail services of PLATINUM members first they have to apply for DIAMOND and then they can go for PLATINUM membership. DIAMOND members can go only for PLATINUM membership and similarly REGULAR.
   Note: While moving from one membership to other keeping sub access level consistent.
5. **View_membership_history** () to display customer details along with all the instances at which his/her membership has been changed.
6. **Priority_based_token_generation ():** all the waiting customers can be added to active list as per their arrival time. If there are more than **'n'**customers are waiting to avail services. A priority based token generation scheme based on arrival time should be implemented.

Note: This function is only for waiting list customer. So it should be executed whenever any active customer in corresponding membership level has completed availing services or session time expires.

7. **Remove_token ():** There is a fixed time to avail services after token generation e.g. 15Min. If the session expires then corresponding token should be removed and hence respective customer should be removed from token list.

Hints :

1. You can maintain separate database for customers. Tree can be used for the same. You can decide what all data needs to be maintained for a customer.
2. Priority queue can be used to implement waiting/active queues.

Problem for R5 Batch:

Multilevel Marketing Problem

A multilevel marketing is a business model in which different levels of distributors exist. A special case of such business model is one where each distributor, known as independent business owner (IBO), may perform two roles- i) may market products directly to potential customers and ii) may mentor other people to become IBOs.IBOs may earn income both from the retail markup on any products they sell personally, plus a performance bonus based on the sales volume they and their downline (IBOs they have sponsored) have generated. Imagine such a business network where one person can make atmost three IBOs. For creating first two IBOs in his chain, the person will get 100 points each but for making the third one, he will earn 200 points. When a new IBO is created in the chain by a mentor, all ancestors of the mentors will also get 10 points irrespective of their level of hierarchy. Again, one can sale 5 different categories of items for which scoring amounts are 10, 20, 30, 40 and 50 points. That is a person scores 10 points for selling a product of first category and so on.

I. Construct a tree to model such a multilevel marketing chain in which all IBOs whose mentor is same, will be represented as children nodes of the mentor. A mentor can have maximum three children. All IBOs whose mentors are under the same mentor will be in the same level in the tree.
II. Define **insert_IBO(ibo, mentor)** function to insert a new node “ibo” in proper position under “mentor” when a new IBO will be created. Each IBO node stores the information regarding IBO_id (it should be generated by the system and should be unique), mentoring_score (i.e., total points scored for insertion of any new IBO in the downline), sale_score(i.e., total points scored on purchase of items)
III. Display the names of the highest scorer (mentor_score+sale_score) at each level.
IV. Find out the IBOs at each level who has the maximum number of descendants.
V. Find if there is any such IBO who has more points than his mentor.
VI. Write a **delete_IBO( )** function for the tree when any IBO will leave. In this situation all children of the IBO who is leaving will become the children of his mentor and the mentoring as well as sale scores are added to that of the mentor.

Problem for R6 Batch:

Image representation problem

A binary image is an image in which intensity value of each pixel is either 0 or 1. The value 0 represents the background (white portion in Fig. 1) and 1 represents the foreground pixel, i.e., actual object pixel (shaded portion in Fig.1). Consider a binary image/matrix of size (n pixels) X (n pixels) [$n=4^m$, where m is an integer]. Divide the entire image space into four equal segments recursively as shown in Fig. 2. The segments are labelled as A, B, C, D. Each segment is again subdivided into four equal segments and the process continues recursively until each segment contains a single pixel. Pixels of each segment are labelled in clockwise direction.

We can decide to store the image in memory as a tree. Each node of the tree has four children. A, B, C, D are the four children of the parent node. Each level of segmentation is represented at the same level nodes in the tree. The internal nodes store the labels. The leaves store the intensity value (0/1). The root is a node representing the complete image/matrix.

0	0	0	0
0	1	1	0
0	0	0	1
0	0	0	0

Fig. 1

A1	A2	B1	B2
A4	A3	B4	B3
D1	D2	C1	C2
D4	D3	C4	C3

Fig. 2

A	B
D	C

Fig. 3

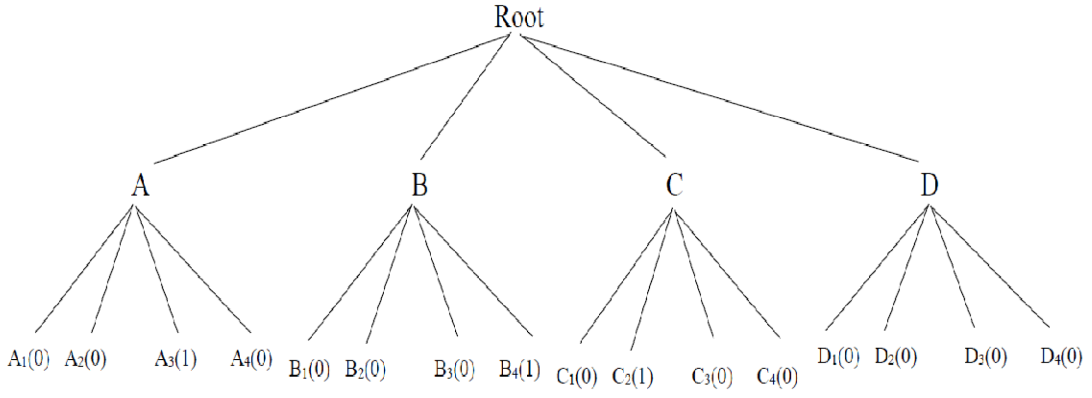

Fig. 4

I. Write a function to implement the tree structure for a given image. Each internal node stores the label of the segment as given in figure 4.. Each leaf node stores the label of the segment, coordinates and pixel intensity value.
Hint : Knowing the value of "n", you can determine the number of levels of the tree.

II. Write a function to find the total number of foreground pixels (value 1) in the tree.

III. Implement an efficient search algorithm to count the number and pairs of disjoint* foreground objects present in the image by traversing the tree.
   a. First consider every pixel as an object. So, disjoint pixels to be found in the original image matrix given.
   b. If we consider an object at any level in the tree as a foreground object, if it contains a pixel 1 inside (i.e. in its subtree), find the number and pairs of foreground objects at every level in the tree.

Note: *Two objects are said to be disjoint if no foreground pixels in the first object is a neighbouring pixel of any foreground pixel in the second object.

IV. Traverse** the tree to display the image in
   a) row wise fashion
   b) column wise fashion
   c) spiral traversal of the given image.
      Spiral traversal for figure 3 is as follows :
      A1, A2, B1, B2, B3, C2, C3, C4, D3, D4, D1, A4, A3, B4, C1, D2

**You can take help of the matrix format of the image, if you want.