

# Big O analysis

Tuesday, March 29, 2022 3:56 PM

## METHOD 1:

```
void MainWindow::on_pushButton_clicked()
{
    ui->listWidget_price->clear();
    ui->listWidget_item->clear();
    ui->listWidget_distance->clear();
    ui->listWidget_name->clear();
    restaurant thisRestaurant;

    // once a restaurant is chosen from the listWidget, its information will be displayed
    QString resName = ui->listWidget->currentItem()->text();
    ui->lineEdit->setText(resName); // display restaurant name

    for (int i = 0; i < restaurantList.size(); i++)
    {
        if (restaurantList[i].getRestaurantName() == resName)
            thisRestaurant = restaurantList[i];

    }

    // display distance to Saddleback
    ui->lineEdit_2->setText(QString::number(thisRestaurant.getDistanceToSaddleback()));

    // display menu (item names and item prices)
    QList<menuItem> resMenu = thisRestaurant.getMenu();
    for (int i = 0; i < resMenu.size(); i++)
    {
        ui->listWidget_item->addItem(resMenu.at(i).itemName);
        ui->listWidget_price->addItem(QString::number(resMenu.at(i).itemPrice));

    }

    // display distance to others
    QVector<distanceInfo> resDistanceList = thisRestaurant.getDistanceList();
    QString toWhichName;
    for (int i = 0; i < resDistanceList.size(); i++)
    {
        toWhichName = GetRestaurantNameUsingQSL(resDistanceList.at(i).toWhich);
        ui->listWidget_name->addItem(toWhichName);
        ui->listWidget_distance->addItem(QString::number(resDistanceList.at(i).distance));

    }
}
```

Big O notation:  $O(n)$

This method displays restaurant name, menu items, distance to Saddleback and distances to others in linear time

## METHOD 2:

```
QString MainWindow::AddApostropheToString(QString restaurantname)
{
    // change to sql query format when string contains apostrophe (need to add one more ' to the query)

    if(restaurantname.contains("'") || restaurantname.contains('\'))
    {
        auto parts = restaurantname.split(QLatin1Char{'\''});
        QString first = parts.at(0);
        QString second = parts.at(1);

        first.chop(1);
        restaurantname = first + "\\'" + second;

    }

    return restaurantname;
}
```

Big O notation:  $O(1)$

This method modifies a string (restaurant name) in constant time

### METHOD 3:

```
void MainWindow::on_deleteMenuItem_clicked()
{
    QString restaurantname = ui->listWidget->currentItem()->text();
    QString MenuItem = ui->listWidget_item->currentItem()->text();
    QString restName = AddApostropheToString(restaurantname);
    int restaurantId = GetRestaurantIDUsingQSL(restName);
    double prices = GetRestaurantPriceUsingQSL(restName, MenuItem);

    restaurant thisRestaurant;
    QString resName = ui->listWidget->currentItem()->text();

    for (int i = 0; i < restaurantList.size(); i++)
    {
        if (restaurantList[i].getRestaurantName() == resName)
        {
            thisRestaurant = restaurantList[i];
        }
    }
    QList<menuItem> resMenu = thisRestaurant.getMenu();
    menuItem eachMenuItem;
    resMenu.removeAt(ui->listWidget_item->currentRow());

    thisRestaurant.setMenu(resMenu);
    restaurantList[restaurantId-1] = thisRestaurant;

    const QString DRIVER("QSQLITE");
    if (QSqlDatabase::isDriverAvailable(DRIVER))
    {
        QSqlDatabase db = QSqlDatabase::addDatabase(DRIVER);
        QString dbPath = QApplication::applicationDirPath() + "/restaurant.sqlite";
        db.setDatabaseName(dbPath);
        db.open();

        QSqlQuery qry(db);
        QString stringQry = "DELETE FROM menu WHERE price = " + QString::number(prices) + " AND restaurantID = " + QString::number(restaurantId) + " AND item = '" + MenuItem + "'";
        qry.prepare(stringQry);
        if (!qry.exec(stringQry))
        {
            qWarning() << "ERROR: Deleting from menu" << qry.lastError().text();
        }

        ui->listWidget_item->currentItem()->setHidden(true);
        ui->listWidget_item->currentItem()->setText("");
        ui->listWidget_price->setCurrentRow(ui->listWidget_item->currentRow());
        ui->listWidget_price->currentItem()->setText("");
        ui->listWidget_price->currentItem()->setHidden(true);
        db.close();
        QString connectionName = db.connectionName();
        db = QSqlDatabase();
        QSqlDatabase::removeDatabase(connectionName);
    }
}
```



Big O notation:  $O(n)$

This method deletes a menu item of the restaurant and update the list and the database in linear time.

# Big O analysis

Tuesday, March 29, 2022 3:56 PM

## METHOD 4 :

```
void customtrip::on_pushButton_6_clicked()
{
    db.close();
    const QString connectionName = db.connectionName();
    db = QSqlDatabase();
    QSqlDatabase::removeDatabase(connectionName);
    hide();
    if(admin == true)
        emit Admin();
    else
        emit backMain();
    if(!ui->lineEdit_totalDistance_3->text().isEmpty())
    {
        QString output;
        output += "The total distance for the trip was ";
        output += ui->lineEdit_totalDistance_3->text();
        output += "\n";
        output += "-----";
        output += "\n";
        output += "The total revenue at each individual restuarant:";
        output += "\n";
        for(int i=0; i<revenueRecords.size(); i++)
        {
            QString name = revenueRecords[i].restaurantName;
            QString amount = QString::number(revenueRecords[i].revenue);
            name = name.leftJustified(30, ' ');
            output += name;
            output += "$";
            output += amount;
            output += "\n";
        }
        output += "-----";
        output += "\n";
        output += "The grand total for the trip was ";
        if(ui->lineEdit_totalSpentTrip_3->text().isEmpty())
            output += "$0.00";
        else
            output += ui->lineEdit_totalSpentTrip_3->text();

        QMessageBox::about(this, "TRIP SUMMARY", output);
    }
}
```



Big O notation:  $O(n)$

This method calculates and outputs the total spending for the trip (sum of total spending of each restaurant visited in linear time.

## METHOD 5:

```
void MainWindow::on_SubmitNew_clicked()
{
    //add new item and price to list
    QListWidgetItem* ItemName = new QListWidgetItem(ui->editNewItem->text());
    QListWidgetItem* ItemPrice = new QListWidgetItem("$" + ui->editNewPrice->text());
    ui->listWidget_item->addItem(ItemName);
    ui->listWidget_price->addItem(ItemPrice);

    //get current restaurant name and add new info to the database
    QString restaurantname = ui->listWidget->currentItem()->text();
    QString restName = AddApostropheToString(restaurantname);
    int restaurantId = GetRestaurantIDUsingQSL(restName);

    restaurant thisRestaurant;
    QString resName = ui->listWidget->currentItem()->text();


    for (int i = 0; i < restaurantList.size(); i++)
    {
        if (restaurantList[i].getRestaurantName() == resName)
        {
            thisRestaurant = restaurantList[i];
        }
    }

    QList<menuItem> resMenu = thisRestaurant.getMenu();
    menuItem eachMenuItem;
    eachMenuItem.itemName = ui->editNewItem->text();
    eachMenuItem.itemPrice = (ui->editNewPrice->text().toFloat());
    resMenu.append(eachMenuItem);
    thisRestaurant.setMenu(resMenu);
    restaurantList[restaurantId-1] = thisRestaurant;

    const QString DRIVER("SQLITE");
    if (QSqlDatabase::isDriverAvailable(DRIVER))
    {
        QSqlDatabase db = QSqlDatabase::addDatabase(DRIVER);
        QString dbPath = QApplication::applicationDirPath() + "/restaurant.sqlite";
        db.setDatabaseName(dbPath);
        db.open();
        QSqlQuery qry(db);

        qry.prepare("INSERT INTO menu VALUES(:restaurantID, :item, :price)");
        qry.bindValue(":restaurantID", restaurantId);
        qry.bindValue(":item", ui->editNewItem->text());
        qry.bindValue(":price", ui->editNewPrice->text());
        if(!qry.exec())
            qWarning() << "ERROR: UPDATING menu" << qry.lastError().text();

        db.close();
        QString connectionName = db.connectionName();
        db = QSqlDatabase();
        QSqlDatabase::removeDatabase(connectionName);
    }
}
```



Big O notation:  $O(n)$

This method adds new menu item to the restaurant, updates the restaurant list and the database in linear time.

## METHOD 6 :

```
// when done button is clicked
void ClosestTrip::on_pushButton_5_clicked()
{
    /*
     * This section of code checks if any item has an accumulative quantity of over 100,
     * if so, a message box pops out and the function ends there
     */
    for(int i=0; i<ui->listWidget_cartItem->count(); i++) n
    {
        QListWidgetItem* temp1 = ui->listWidget_cartItem->item(i);
        QString checkingRestaurant = temp1->text();
        int count = 0;
        for(int j=0; j<ui->listWidget_cartItem->count(); j++)
        {
            QListWidgetItem* itemName = ui->listWidget_cartItem->item(j);
            if(itemName->text() == checkingRestaurant)
            {
                count += ui->listWidget_2->item(j)->text().toInt();
            }
        }
        if(count > 100)
        {
            QMessageBox::critical(this, "\\Quantity Limit Exceeded\\", "We're sorry, but we do not allow ordering more than 100 of the same menu item. Please remove items accordingly. Thank you.");
            return;
        }
    }

    float totalSpentOnRestaurant = 0.0;
    for(int i = 0; i < ui->listWidget_cartPrice->count(); i++) n
    {
        QListWidgetItem* item = ui->listWidget_cartPrice->item(i);
        totalSpentOnRestaurant += item->text().toFloat() * ui->listWidget_2->item(i)->text().toInt();
    }

    ui->lineEdit_totalOnRest->setText("$" + QString::number(totalSpentOnRestaurant));

    for(int i=0; i<revenueRecords.size(); i++) n
    {
        if(revenueRecords.at(i).restaurantName == ui->name->text())
            revenueRecords[i].revenue = totalSpentOnRestaurant;
    }

    totalSpendingOnTrip += totalSpentOnRestaurant; // add total spent on this restaurant to total spent on trip
    ui->lineEdit_totalSpentTrip->setText("$" + QString::number(totalSpendingOnTrip)); // update/display total spent on trip

    ui->listWidget_cartItem->clear();
    ui->listWidget_cartPrice->clear();
    ui->listWidget_2->clear();
    ui->listWidget->currentItem()->setFlags(ui->listWidget->currentItem()->flags() & ~Qt::ItemIsEnabled);
    ui->listWidget_menu->clear();
    ui->listWidget_price->clear();
    ui->name->clear();
    ui->pushButton_5->setDisabled(true);
}
```

Big O notation:  $O(n^2)$

This method checks if any cart item has the accumulative quantity over 100 and displays total spending on trip in quadratic time.