Project 1 Big-Oh Discussion

Team: Go Big O Go Home
Members: Matt Jang, Lamson Bui, Angaar Hamid, Suqi Hu

- **BeginTrip() -** The method below is **O(n)** because the method will recursively determine the closest distance from a starting college and an ending college until the vector -which stores the list of selected colleges- is empty. Thus, the method is largely dependent on the size of the collegesVector until the entire list of colleges has been accounted for.

```
// BeginTrip() - Will recursively order the trip in terms of efficiency
void DBManager::BeginTrip(QString startingCollege, QVector<QString> collegesVector, double
&totalDistance)
{
  QSqlQuery qry;
  collegesVector.pop_front(); // pops front of the vector

  // Base case: if vector is empty, exit
  if(collegesVector.isEmpty()) {
     return;
  }

  // General case
  QString closestCollege;
  qry.prepare("Select endingCollege from CollegeDistances where distanceBetween = "
           "(Select min(distanceBetween) from CollegeDistances where startingCollege =
'"+startingCollege+"' and "
           "endingCollege not in (Select CollegeName from AlreadyVisitedColleges) and
endingCollege in "
           "(Select Queue from TourData)) and startingCollege = '"+startingCollege+"';");

  // Stores the closest college into the string
  if(!qry.exec()) {
     qDebug() << "Can't execute closes college sql statement!";
  }
  if(qry.next()) {
     closestCollege = qry.value(0).toString();
   }

  // Will accumulate distance traveled from startingCollege to closestCollege
  qry.prepare("Select distanceBetween from CollegeDistances where startingCollege =
'"+startingCollege+"' and endingCollege = '"+closestCollege+"';");
  if(!qry.exec()) {
     qDebug() << "Can't compute total distance!";
  }
  double distanceTraveled;
  if(qry.next()) {
    distanceTraveled = qry.value(0).toDouble();
  }
  totalDistance += distanceTraveled;
  qDebug() << totalDistance;


  // Insert into AlreadyVisitedColleges table the closest college
  qry.prepare("Insert into AlreadyVisitedColleges(CollegeName)
VALUES('"+closestCollege+"');");
  if(!qry.exec()) {
     qDebug() << "Can't insert closestCollege into AlreadyVisitedColleges";
  }

  // changes startingCollege to be closestCollege, and calls the function again
  startingCollege = closestCollege;
  BeginTrip(startingCollege, collegesVector, totalDistance);
}
```

- **DepartButton()** - The method below is **O(n)** as the vector which stores all the selected colleges requires **O(n)** allocation time. The vector will continually append the colleges from a database which requires **n** space.

```
    // DepartButton() - Will officially start the trip
void MainWindow::on_DepartButton_clicked()
{
    // selects colleges from AlreadyVisitedColleges table
    QSqlQuery qry;
    qry.prepare("Select CollegeName from AlreadyVisitedColleges;");

    if(!qry.exec()) {
        qDebug() << "Can't add sorted colleges to vector!";
    }

    // appends newly sorted colleges into the vector
    selectedCollegesVector.clear();
    while(qry.next()) {
        selectedCollegesVector.append(qry.value(0).toString());
    }
```

```
  // If user doesn't have any colleges, then an error message will appear
  if(selectedCollegesVector.isEmpty()) {
    QMessageBox::warning(this, "Warning", "Please select a college");
    return;
  }

  ui->stackedWidget->setCurrentWidget(ui->CampusPage);          // changes page to
CampusPage

  // sets collegeName label to be first college's name
  QString firstCollege = selectedCollegesVector.at(0);
  ui->collegeNameLabel->setText(firstCollege);

  // sets collegeDescription for text browser
  ui->collegeDescriptionTextBrowser->setText(GetCollegeDescription(firstCollege));

  // PROCESSING - Sets picture path depending on firstCollege
  QString pixelPath = GetPicturePath(firstCollege);
  QPixmap pix(pixelPath);
  int w = ui->collegePicturesLabel->width();
  int h = ui->collegePicturesLabel->height();

ui->collegePicturesLabel->setPixmap(pix.scaled(w,h,Qt::KeepAspectRatio,Qt::SmoothTransfor
mation));

  // Sets the souvenir table view for the corresponding college
  ui->souvenirTableView->setModel(databaseObj.LoadSouvenirsByCollege(firstCollege, false));

  this->currentPrice = 0; // sets current price of each college equal to 0
  this->totalPrice = 0;   // sets total price equal to 0
  ui->priceLCDNumber->display("0");

}
```

- **on_nextCollegeButton_clicked() -** The method below is **O(1)** as the operations performed do not require any loops and only perform basic operations. For example, pop_front() and front() methods of a vector are used which only take constant time.

```
// Will go to next college during the tour trip
void MainWindow::on_nextCollegeButton_clicked()
{
  // pops front element in vector, sets currentPrice equal to 0 (recalculate price for each college)
  selectedCollegesVector.pop_front();
  currentPrice = 0;
  ui->priceLCDNumber->display(0);

  // if vector is empty, then tour is finished, will go back to front page
  if(selectedCollegesVector.isEmpty()) {
     on_backButton_7_clicked();  // goes back to first page
     QMessageBox::information(this, "Information", "Thank you for touring with us. Your total is :
$ " + QString::number(totalPrice));
     totalPrice = 0; // resets totalPrice back to 0

     // Deletes TourData table and AlreadyVisitedColleges table
     Delete_Tour_Data();
     DeleteAlreadyVisitedCollegesTable();
  }
  else {
     // Gets the nextCollege from vector
     QString nextCollege = selectedCollegesVector.front();
     ui->collegeNameLabel->setText(nextCollege);

     // sets collegeDescription for text browser
     ui->collegeDescriptionTextBrowser->setText(GetCollegeDescription(nextCollege));

     // PROCESSING - Sets picture path depending on college
     QString pixelPath = GetPicturePath(nextCollege);
     QPixmap pix(pixelPath);
     int w = ui->collegePicturesLabel->width();
     int h = ui->collegePicturesLabel->height();

ui->collegePicturesLabel->setPixmap(pix.scaled(w,h,Qt::KeepAspectRatio,Qt::SmoothTransfor
mation));

     // Sets the souvenir table view for the next college
     ui->souvenirTableView->setModel(databaseObj.LoadSouvenirsByCollege(nextCollege,
false)); }
```

- **GetSouvenirPrice() -** The method below takes **O(1)** time as the method searches for the souvenir at a specific college with the addition of the college's name and the souvenir's name. Thus, the method does not need to search the entire table or list of colleges to return the souvenir's price.

```
  // GetSouvenirPrice() - Returns price of the corresponding item at a given college
double DBManager::GetSouvenirPrice(QString collegeName, QString itemName)
{
  double price;
  QSqlQuery qry;

  qry.prepare("select printf(\"%.2f\", sum(cost)) as \"Price\" from Souvenirs where college =
'"+collegeName+"' "
             "and traditionalSouvenirs = '"+itemName+"';");

  if(!qry.exec())
  {
     qDebug() <<"error Loading values to db" << endl;

  }

  if(qry.next()) {
     price = qry.value(0).toDouble();
  }
  return price;
}
```

- **loadAlreadyVisitedCollegesTable()** - This method requires **O(n)** time as the method needs to iterate through the entire list of colleges stored in a database table.

```
// loadAlreadyVisitedCollegesTable() - Returns a QSqlQueryModel consisting of information
from AlreadyVisitedColleges table
QSqlQueryModel *DBManager::loadAlreadyVisitedCollegesTable()
{
  QSqlQueryModel* model = new QSqlQueryModel();

  QSqlQuery qry;
  qry.prepare("SELECT CollegeName from AlreadyVisitedColleges;");

  if(!qry.exec())
  {
     qDebug() <<"error Loading values to db" << endl;

  }
```

```
    model->setQuery(qry);

    return model;
}
```