Due: Friday, 11/8/2019 before 5:00PM

Total points = 50

1 Overview

In this assignment, you will process map data from OpenStreetMap for the City of Buffalo to build a graph of streets that form an intersection. Using this graph, we will then be able to determine the shortest distance between two properties in Buffalo and produce the "driving directions" to get between them.

2 Objectives

In this assignment you will:

- Load map data in XML format.
- Synthesize multiple data sources to solve
- Perform simple graph algorithms to produce driving directions.
- Use a simple graph data structure.

3 Useful Resources

In addition to the course lecture notes, the following should also help with completing the assignment tasks:

- Mark Lewis Graphs + Graph Traversals (https://www.youtube.com/playlist?list=PLLMXbkbDbVt_1iNav8jWDXkeReYOeg0qz)
- Graph Traversals (Section 15.3 p473-481)
- Scala Cookbook - XML: https://alvinalexander.com/scala/how-to-extract-data-from-xml-nodes-in-scala
- Scala-XML Getting Started: https://github.com/scala/scala-xml/wiki/Getting-started

Note that for the XML processing, all we need to do is load XML data and access the data. No modifications need to be performed and no manual parsing should be done.

4 Instructions

Complete the following programming problems and submit your answers to Autolab via the appropriate submission location. **Expect this section to take 10-12 hours of reviewing documentation, planning your approach, and writing the code.**

Problem 1. (10 points)
In `MapUtilities` found in `MapUtilities.scala`, complete the method `loadIntersectionIDs` that returns a mutable `Set[String]` holding the `id` of each `node` found within the given XML input file.

- File format: valid XML containing all `node` items as a child of an `osm` tag.
- Hint: use `scala.xml.XML` to process the XML file for you.

For example:

```
<node id="111296042" lat="42.8309570" lon="-78.8492830"/>
<node id="111296048" lat="42.8316410" lon="-78.8468080"/>
<node id="279658800" lat="42.8331645" lon="-78.8529137"/>
<node id="354586177" lat="42.8337992" lon="-78.8538948"/>
```

Here we have 4 nodes so these 4 ids: `111296042`, `111296048`, `279658800`, `354586177` should each appear in the output set.

Problem 2. (10 points)
In `MapUtilities`, complete the method `loadMapInfo` that returns a mutable `Map[String,Set[String]]` holding the mapping of each node `id` to the set of street names where that node is found.

- File format: valid XML containing, among other things, `way` items as children of an `osm` tag.
- Each node in the file is a point along a some street. These nodes are used to mark the start-/end- points of a road, intersections, and more.
  - Some nodes correspond to intersections. In this case we should find multiple ways linked to the same node.
  - Some nodes correspond to sidewalks/alleys/driveways/etc. so you may find that only one street maps to a given node id.
- Only the data within `way` elements is of interest for this input file.
  - The `tag` elements represent `key->value` mappings by holding values under their attributes `k` (for key) and `v` (for value).
  - The `tag` elements with the attribute `k` (key) as `tiger:name_base` holds the street name as the value of the attribute `v`.
  - Any `way` element that is not a street should be ignored.
    - The `way` does not correspond to a street if no child `tag` element contains an attribute `k` with the value `tiger:name_base`.
  - For each child `nd` element, extract the `ref` attribute (an id). Store the street name into the set corresponding to the id in the output map.
- Hint: use `scala.xml.XML` to process the XML file for you.

haohuafe@buffalo.edu - Oct 27, 2019, 10:58:42 PM EDT

For example:

```
<way id="12271118" version="4" timestamp="2015-12-10T02:29:12Z"
 changeset="35857848" uid="599436" user="zeromap">
  <nd ref="111226479"/>
  <nd ref="111226482"/>
  <tag k="highway" v="residential"/>
  <tag k="maxspeed" v="30 mph"/>
  <tag k="name" v="Walter Street"/>
  <tag k="tiger:cfcc" v="A41"/>
  <tag k="tiger:county" v="Erie, NY"/>
  <tag k="tiger:name_base" v="Walter"/>
  <tag k="tiger:name_type" v="St"/>
  <tag k="tiger:zip_left" v="14210"/>
  <tag k="tiger:zip_right" v="14210"/>
</way>
```

This `way` element corresponds to Walter St. The name you would extract is `"Walter"` from

`<tag k="tiger:name_base" v="Walter"/>`.

In the output mapping `"Walter"` should be added to the sets under `111226479` and `111226482`. If you look through the dataset, you'll also find find the only other element that `111226479` is associated with is Quinn St:

```
<way id="12284253" version="4" timestamp="2015-12-10T02:28:02Z"
 changeset="35857848" uid="599436" user="zeromap">
  <!-- ... -->
  <nd ref="111226479"/>
  <!-- ... -->
  <tag k="tiger:name_base" v="Quinn"/>
  <tag k="tiger:name_type" v="St"/>
</way>
```

This means in the output map you return, you would have the pair:

`111226479 -> Set("Quinn", "Walter")`

Problem 3. (10 points)

In `MapUtilities`, complete the method `buildIntersectionGraph` that returns a `StreetGraph` that contains an edge between any two streets that intersect, and no edge otherwise (based on the input dataset).

Now that we have been able to load the data from OpenStreetMaps and the intersection data for Buffalo, we now want to build a graph to represent this information. In particular, the graph should have an edge between two street names if they intersect. From our previous example, we know that `Walter` and `Quinn` streets intersect in Buffalo (if you don't believe it, go on Google maps and search "Walter St at Quinn St Buffalo"). This means that in our graph there should be an edge `(Walter,Quinn)` and another edge `(Quinn, Walter)`.

- In the given node ID sequence, it's possible there is no corresponding intersection contained in the map info or there is only one street listed at the intersection. These IDs can be ignored.

- It is possible an intersection connects multiple streets. In this case, edges should be added appropriately.

Problem 4. (10 points)
In `MapUtilities`, complete the method `computeFewestTurns` that returns an `Int` corresponding to the fewest number of streets traveled to get from one `TaxEntry` to another.

- If they are on the same street the result should be 0.
- If it is not possible to reach the other property, return -1.

Problem 5. (10 points)
In `MapUtilities`, complete the method `computeFewestTurnsSeq` that returns a `Seq[String]` giving the streets to take corresponding to the fewest number of streets traveled to get from one `TaxEntry` to another.

- The result should always start with the starting street and end with the ending street.
- If they are on the same street the result should contain only 1 street.
- If it is not possible to reach the other property, return an empty sequence.

Note that there may be multiple valid sequences, but you only need to output 1 correct one.

4.1 Getting Started

5(a) Download the handout and the required data files.

5(b) Enter your ubit and person number in your submission file.

5(c) Work on problems in the order they are assigned.

5 Allowed Header/Library Usage

- You may use Arrays/Lists/ArrayBuffers/ListBuffers as well as the Sets and Maps as necessary.
- You may use scala.xml.XML
- You may use Stacks or Queues.

No other headers are allowed.

6 Submission

For each section (PA3 Programming), you will be allowed 5 submissions to each, without penalty. Starting from the 6th submission, you will receive

- a 5 points per submission deduction from your score on the respective assignment.

Note: your score is what you receive on your latest submission. If you receive a score and then resubmit, even if you receive a lower score/0 points, that will be your score for the assignment.

Also note: if you submit early/late, the bonus/penalty would be awarded against the entirety of the assignment, not just the single part you submitted early/late. The timestamp for your submission is that of the latest part submitted.

haohuafe@buffalo.edu - Oct 27, 2019, 10:58:42 PM EDT

Creating Your Submission

Your submission should be a single file: `MapUtilities.scala`. No other files should be provided for grading.

7 Late Policy

The policy for late submissions on assignments is as follows:

- More than 5 days before the deadline: +5 points + 100% of what you earn on your best submission.
- Up to 5 days before the deadline: +1 point per day + 100% of what you earn on your best submission.
- On the deadline: 100% of what you earn on your best submission.
- One day late: 25% point deduction.
- Two days late: 50% point deduction.
- $>$ 2 days late: 0 points.

A day is considered as a 24 hour period counting from the deadline time. From this policy, you will see that all assignments must be submitted within two days past the assigned deadline. The date of submission is that of your latest submission made, even if this is not your highest scoring submission. For assignments with multiple component submissions, only 1 penalty will be assessed based on the file submitted last. If a staggered deadline is given (e.g., two components due one week apart), the earlier deadline will be a hard deadline and no late submissions will be accepted for the first component.

You will have the ability to use three grace days throughout the semester, and at most two per assignment (since assignments are not be accepted beyond two days late). Using a grace day will negate the 25% point penalty for one day of late submission, but will not allow you to submit more than two days late. Please plan accordingly. You will not be able to recover a grace day if you decide to work late and your score was not sufficiently higher. **Grace days are automatically applied** to the first instances of late submissions, **and are non-refundable**. For example, if an assignment is due on a Friday and you make a submission on Saturday, you will automatically use a grace day, regardless of whether you perform better or not. **Be sure to test your code before submitting**, especially with late submissions, **in order to avoid wasting grace days**.

Keep track of the time if you are working up until the deadline. Submissions become late after the set deadline. Keep in mind that **submissions will close 48 hours after the original deadline** and you will no longer be able to submit your code after that time.

8 AI Policy Overview

As a gentle reminder, please re-read the academic integrity policy of the course. I will continue to remind you throughout the semester and hope to avoid any incidents.

8.1 What constitutes a violation of academic integrity?

These bullets should be obvious things not to do (but commonly occur):

- Turning in your friend's code/write-up (obvious).

- Turning in solutions you found on Google with all the variable names changed (should be obvious). This is a copyright violation, in addition to an AI violation.
- Turning in solutions you found on Google with all the variable names changed and 2 lines added (should be obvious). This is also a copyright violation.
- Paying someone to do your work. You may as well not submit the work since you will fail the exams and the course.
- Posting to forums asking someone to solve the problem.

  Note: Aggregating every [stack overflow answer|result from google|other source] because you "understand it" will likely result in full credit on assignments (if you aren't caught) and then failure on every exam. Exams don't test if you know how to use Google, but rather test your understanding (i.e., can you understand the problems to arrive at a solution on your own). Also, other students are likely doing the same thing and then you will be wondering why another person that you don't know has your solution.

You should know that seeking solutions to the assignment does not fall under solving the problem yourself. Things that may not be as obvious:

- Working with a tutor who solves the assignment with you. If you have a tutor, please contact me so that I may discuss with them what help is allowed.
- Sending your code to a friend to help them. **If another student uses/submits your code, you are also liable and will receive the same punishment**.
- Joining a chatroom for the course where someone posts their code once they finish, with the honor code that everyone needs to change it in order to use it.
- Reading your friend's code the night before it is due because you just need one more line to get everything working. It will most likely influence you directly or subconsciously to solve the problem identically, and your friend will also end up in trouble.

The assignments should be solved individually with only assistance from course staff and allowed resources. You may discuss and help one another with technical issues, such as how to get your compiler running, etc. It is not acceptable that you both worked together and have nearly identical code. If that is going to be a problem for you, don't solve the problems in that close of proximity.

8.2 What collaboration is allowed?

There is a gray area when it comes to discussing the problems with your peers and **I do encourage you to work with one another to solve problems**. That is the best way to learn and overcome obstacles. At the same time you need to be sure you do not overstep and not plagiarize. Talking out how you eventually reached the solution from a high level is okay:

> I used a stack to store the data and then looked for the value to return.

but explaining every step in detail/pseudocode is not okay:

> I copied the file tutorial into my code at the start of the function, then created a stack and pushed all of the data onto the stack, and finished by popping the elements until the value is found and use a return statement.

The first example is OK but the second is basically a summary of your code and is not acceptable, and remember that you shouldn't be showing any code at all for how to do any of it. Regardless of where you are working, you must always follow this rule: **Never come away from discussions with your peers with any written work, either typed or photographed, and especially do not share or allow viewing of your written code.**

If you have concerns that you may have overstepped or worked closely with someone, please address me prior to deadlines for the assignment. We will address options at that point.

8.3 What resources are allowed?

With all of this said, please feel free to use any [files|examples|tutorials] that we provide directly in your code (with proper attribution). Feel free to directly use anything from lecture or recitations. You will never be penalized for doing so, but must always provide attribution/citation for where you retrieved code from. Just remember, if you are citing an algorithm that is not provided by us, then you are probably overstepping.

More explicitly, you may use any of the following resources (with proper citation/attribution in your code):

- Any example files posted on the course webpage (from lecture or recitation).
- Any code that the instructor provides.
- Any code that the TAs provide.
- Any code from the tour of Scala (https://docs.scala-lang.org/tour/tour-of-scala.html)
- Any code from Scala Collections (https://docs.scala-lang.org/overviews/collections-2.13/introduction.html)
- Any code from Scala API (https://www.scala-lang.org/api/2.13.0/)
- Additional references may be provided as the semester progresses, but only those provided publicly by course staff are allowed to be utilized. These will be listed on Piazza under Resources.

Omitting citation/attribution will result in an AI violation (and lawsuits later in life at your job). This is true even if you are using resources provided.

Lastly, **if you think you are going to violate/have violated this policy, please come talk to me ASAP so we can figure out how to get you on track to succeed in the course**. This policy on assignments is here so that you learn the material and how to think yourself. There is no benefit to submitting solutions (which likely exist in some form).

haohuafe@buffalo.edu - Oct 27, 2019, 10:58:42 PM EDT