BRIDGES Workshop Agenda
TEMPLE UNIVERSITY CENTER CITY
Room 325
Philadelphia, PA 19102
June 6, 2018

8.30-9.00	Breakfast
9.00	Introductions
9.20	BRIDGES Overview
9.40	Workshop Overview
9.50	BRIDGES Setup
10.00	Break (Refreshments, Continue BRIDGES Setup for attendees)
10.30	BRIDGES Basics (3 examples)
	(a) BRIDGES Account/Credentials, 10 element array Example
	(b) Acquiring External Data (Singly linked list of actors/movies)
	(c) Visual Attributes/Styling (Singly Linked list - Earthquake Magnitude/Scale/Color)
12.00pm	Lunch (on your own - we will go somewhere nearby)
1.30pm	Exercise 1 (Will pick 1 out of 3 possible exercises)
2.30pm	Break
2.40pm	Exercise 2 (Will pick 1 out of 3 possible exercises)
3.30pm	PR Pitch (Invite partners to collect feedback)
3.40pm	Workshop Survey
	https://unccpsych.az1.qualtrics.com/jfe/form/SV_bgxw85Ztuirjltz)
3.50pm	Discussion/Direct Feedback
4.10pm	Forms/Paperwork
4.20pm	Adjourn

1. **Location.** Please note the workshop **is not on the main Temple university campus, but Temple University Center City Building, (1515 Market St, Philadelphia, PA 19102) which is in downtown Philadelphia** Maps and Directions: https://www.temple.edu/tucc/about/maps-directions.asp

2. **Parking.** Please see the following link for discounted parking (we will reimburse you for parking as part of stipend). Temple University Center City Parking: https://www.temple.edu/tucc/about/parking.asp

Assignment 0 - SETUP

Goals

The purpose of this assignment is to

1. Create an account on the BRIDGES server.
2. Use your BRIDGES credentials in code.
3. Upload a simple visualization to BRIDGES.
4. Check if it works.

You will generate a visualization that looks like that!

Create an account on the BRIDGES server

1. Go to http://bridgesuncc.github.io/.
2. Click on "Login" (at the upper right corner).
3. Click on "Sign Up!" (at the bottom of the "Log in!" box).
4. Fill out form (Email is only used to recover password).
5. Go to the "Profile" page (link at the top right).
6. Note the "API Sha1 Key"; you will need it in the next part.

Programming part

Task

Visualize an array of 10 elements where each element stores a square number.

Using credentials

1. Open your scaffolded code.
2. Plug in your credentials.
3. Compile and run the code.
4. Follow the link and check that you can see an array of 2 elements.

Changing the array

1. Change the array size to 10.
2. Using a `for` loop, initialize each array entry to store a square number (0, 1, 4, ...).
3. Compile and run the code.
4. Follow the link and check that you can see an array of square numbers.

Help

for Java

Array documentation

Element documentation

for C++

Array documentation

Element documentation

Assignment 1 - List IMDB

Goals

The purpose of this assignment is to learn to

1. Access remote data through BRIDGES.
2. Manipulate a linked list.

You will generate a visualization that looks like that!

Programming part

Task

Build a linked list containing each (actor,movie) pair that appears in the Actor Movie IMDB data set.

Basic

1. Open your scaffolded code.
2. Plug in your credentials.
3. Compile and run the code and observe the basic linked list.

Build Actor Movie linked list

1. Change `SLelement` to be a list of `ActorMovieIMDB`.
2. For each entry in the set of actor movie, create an `SLelement` to store it with an explicit label.
    1. We recommend you add each new entry to the head of the list for simplicity.
3. Compile and run the code and observe the linked list of (actor,movie) pairs.

Help

for Java

SLelement documentation

Element documentation

ActorMovieIMDB documentation

for C++

SLelement documentation

Element documentation

ActorMovieIMDB documentation

Assignment 2 - List Earthquake

Goals

The purpose of this assignment is to learn to

1. Change the style of nodes.
2. Use the earthquake data.

You will generate a visualization that looks like that!

Programming part

Task

Style a linked list of earthquake records.

Node styling

1. Open your scaffolded code.
2. Plug in your credentials.
3. All the styling will be done in `setProperties`.
4. Get the `EarthquakeUSGS` object from the `SLelement`.
5. Get the `ElementVisualizer` object from the `SLelement`.
6. Change the element size based on the earthquake magnitude.
7. Give the element a different shape if the earthquake is in Hawaii, and yet another shape if it is in Alaska.

Help

for Java

SLelement documentation

Element documentation

ElementVisualizer documentation

EarthquakeUSGS documentation

for C++

SLelement documentation

Element documentation

ElementVisualizer documentation

EarthquakeUSGS documentation

Assignment 3 - Graph Bacon Number

Goals

The purpose of this assignment is to learn to

1. Use the IMDB Actor Movie graph.
2. Compute BFS on that graph.
3. Highlight a shortest path in the graph.

You will generate a visualization that looks like that!

Programming part

Task

Highlight the shortest path between two actors in a Movie Actor graph.

Getting Started

1. Open your scaffolded code.
2. Plug in your credentials.
3. Change the style of nodes Cate_Blanchett and Kevin_Bacon_(I), directly attached nodes, and directly attached edges.
4. Compile, run, and visualize.

Perform BFS

1. Write a BFS traversal in `getBaconNumber` that keeps track of parent information. Here is the algorithm:

```
BFS(G=(V,E), root)
  forall v in V
    mark[v] = false;
  mark[root] = true;
  queue.push(root);
  while (! queue.empty() )
    v = queue.pop();
    for (u in neighboor(v))
      if (mark[u] == false)
        mark[u] = true;
        parent[u] = v;
```

2. We recommend using a built-in associative array for storing parents, such as Java's `HashMap` or C++'s `std::unordered_map`.
3. We recommend using a built-in queue, such as Java's `ArrayDeque` or C++'s `std::queue`.

Style the BFS path

1. Start from the Cate_Blanchett node.
2. Color the current node red and make it bigger.
3. Style the edge from the current node to its parent. Make it red and bigger.
4. Go to the parent node and go back to 2 until Kevin_Bacon_(I) has been reached.

Help

for Java

ArrayDeque documentation

HashMap documentation

Element documentation

GraphAdjListSimple documentation

ElementVisualizer documentation

LinkVisualizer documentation

ActorMovieIMDB documentation

for C++

std::queue documentation

std::unordered_map documentation

Element documentation

GraphAdjList documentation

ElementVisualizer documentation

LinkVisualizer documentation

ActorMovieIMDB documentation

Assignment 4 - Graph Earthquake

Goals

The purpose of this assignment is to learn to

1. Access and manipulate remote data through BRIDGES.
2. Manipulate a `GraphAdjList` object.
3. Display a location on a map.

You will be building a visualization that looks like that!

Programming part

Task

Grab recent earthquake data and build a graph representing the locations of the 100 strongest earthquakes.

Basic

1. Open your scaffolded code.
2. Plug in your credentials.
3. Get the most recent 10,000 earthquakes.
4. Only retain the 100 highest magnitude earthquakes.

Place Earthquakes on the map

1. Create a graph where each earthquake is a vertex.
2. Add no edges for now.
3. Pin earthquakes at their longitude and latitude.
4. Tweak the appearance of vertices if you want (e.g., use a different symbols for earthquake in Hawaii or Alaska).
5. Compile, run, and visualize.

Build a graph based on distances

1. For each pair of earthquakes:
   1. Compute the distance using `calcDistance`.
   2. If the earthquakes are closer than 500km, add an edge between them.
2. Compile, run, and visualize.

Show just the graph

1. Deactivate the map overlay (already done in the scaffolding).
2. Unpin the vertices by setting their location to infinity.
3. Compile, run, and visualize.

Help

for Java

Element documentation

GraphAdjListSimple documentation

GraphAdjList documentation

ElementVisualizer documentation

EarthquakeUSGS documentation

Bridges class documentation

for C++

Element documentation

GraphAdjList documentation

ElementVisualizer documentation

EarthquakeUSGS documentation

DataSource documentation

Assignment 5 - BST Earthquake Data

Goals

The purpose of this assignment is to learn to

1. Use (near) real-time collected earthquake data with a binary search tree.
2. Insert earthquake data into the nodes of a binary search tree, keyed on quake magnitude.
3. Highlight quakes by ranges and traversal algorithms in the BST

You will generate a visualization that looks like that!

Programming part

Task

Highlight the shortest path between two actors in a Movie Actor graph.

Getting Started

1. Open your scaffolded code.
2. Plug in your credentials.
3. Retrieve the earthquake data (start with a few hundred), then visualize.
4. Filter them to take out the smallest quakes, visualize.
5. Look at `BST` file. Write a simple recursive algorithm to find and highlight quakes by their magnitudes.
6. Optional exercises: additional algorithms, mark the insertion/find path in a BST, scale node size by magnitude, etc.

Help

for Java

List Documentation

BSTElement documentation

Element documentation

ElementVisualizer documentation

LinkVisualizer documentation

EarthquakeUSGS documentation

for C++

std::vector documentation

Element documentation

BSTElement documentation

ElementVisualizer documentation

LinkVisualizer documentation

EarthquakeUSGS documentation

Assignment 6 - Square Fill

Goals

The purpose of this assignment is to learn to

1. Manipulate a ColorGrid object.
2. Generate colorful squares

You will generate a visualization that looks like that!

Programming part

Task

You will be drawing Square Fillusing the following logic: (examples provided below.)

1. Start with an empty canvas.
2. Generate a random point that has not been painted yet. Paint it, that is a square of dimension 0.
3. If the outer layer of the square has not been painted yet (aka, if none of the pixel of the outer layer has been painted)
4. Paint it and go to the next layer.
5. Keep painting layers until you reach the border of the image, or one of the layer has a painted pixel.
6. Pick a new random point.

Basic

1. Open your scaffolded code.
2. Plug in your credentials.
3. Complete the TODOs in the scaffoled code.
4. Run and visualize your code

Build Square Fill

1. Plug in your credentials

2. Observe how the code initizes the `ColorGrid`. The code provides a variable that represents a pixel is free, which is embedded into the `Color` objects alpha value. This allows for a much quicker check later on to see if a pixel is free, you may refactor this if you do not wish to use the alpha value for this purpose.

3. Provide the code necessary to generate a random point on the grid, check that point to make sure it is free, and then set that location to some random color.

4. An example snippet is provided in the next TODO that gives an example of how to set up some int's to keep track of each layers boundaries. Ensure that these points are not out of bounds before continuing.

5. In the first for loop, provide the logic for checking each point on the current layer to make sure it is free, if not break the loops to generate a new point on the grid to start from.

6. If the current layer has no collisions, generate a random color and then begin filling in

the points along the current layer with the generated color.

7. Run and visualize the code.

Help

for Java

ColorGrid documentation

Color documentation

Bridges class documentation

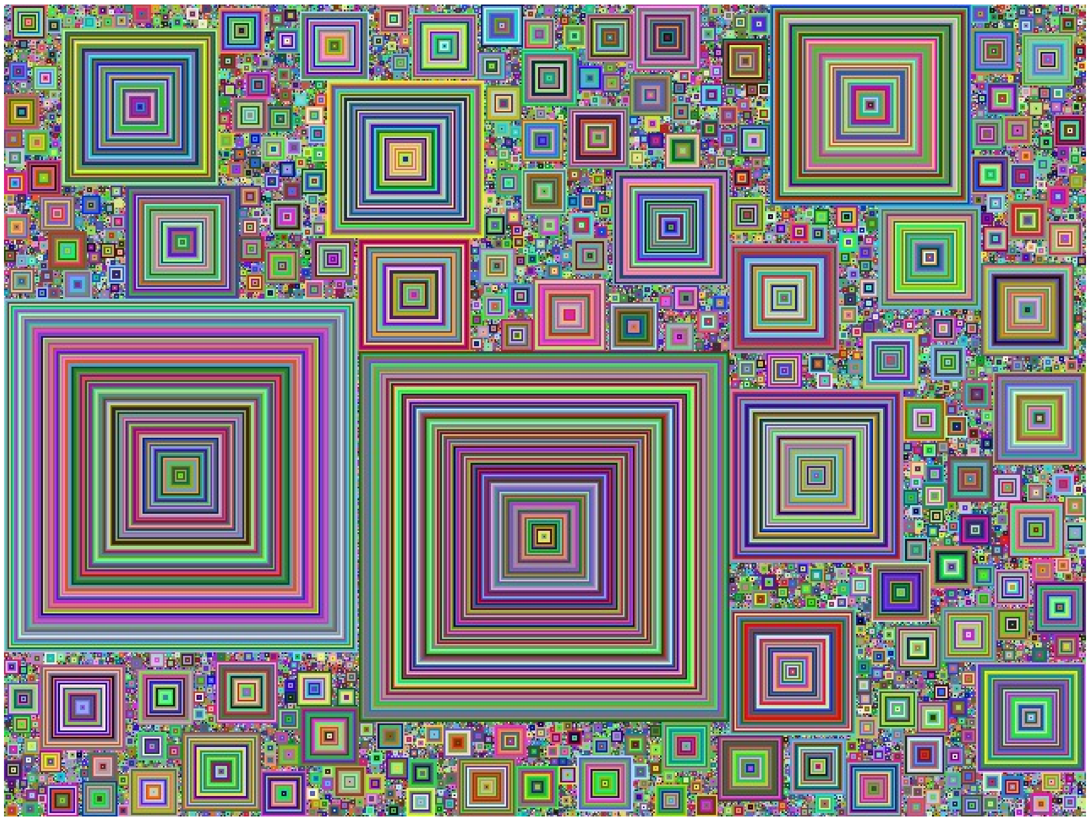
for C++

ColorGrid documentation

Color documentation

DataSource documentation

Sample Output



Assignment 7 - Grid Lyrics

Goals

The purpose of this assignment is to learn to

1. Access lyrics data through BRIDGES.
2. Manipulate a `ColorGrid` object.
3. Show repetition patterns in songs.

You will generate a visualization that looks like that!

Programming part

Task

In this assignment, the objective is to pull a song from Bridges, split the lyrics into individual words, and compare each word against every other word to check for repetition.

From these lyrics, you will be building a matrix, or a `ColorGrid` in this case, where every row and every column represents a sequential word in the song's lyrics.

Upon finding repetition, you will be setting the pixel at that location to a color of your choice at that point in the grid.

Basic

1. Open your scaffolded code.
2. Plug in your credentials.
3. Complete the TODO's.
4. Run and visualize the code.

Build a ColorGrid

1. Plug in your credentials.
2. Think of any song which contains words.
3. Query `Bridges` for said `Song`. For example

- in Java

```
Song mySong = Bridges.getSong("My Favorite Song", "Optional Artist String");
String lyrics = mySong.getLyrics();
```

- in C++

```
Song mySong = DataSource::getSong("My Favorite Song", "Optional Artist String");
auto lyrics = mySong.getLyrics();
```

4. Pass these lyrics through the provided helper function, which will clean up and split the lyrics into an array of squeaky clean `Strings`.

5. Initialize a `ColorGrid` with the dimensions the size of the array returned from the helper function.

6. Iterate over the split lyrics array, checking to see if there is any repetition. For example, if word 1 is the same as the word 6, you would color the pixel at (1, 6), and later on at (6,

1).

7. After filling out your grid, set it as the data structure on your Bridges object, and run the code.

Help

for Java

ColorGrid documentation

Color documentation

Song documentation

Bridges class documentation

for C++

ColorGrid documentation

Color documentation

Song documentation

DataSource documentation

Assignment 8 - Priority Queue Book

Goals

The purpose of this assignment is to learn to

1. Access Shakespear's work with BRIDGES
2. Write your own tree based data structure: A Binary Min-Heap

You will generate a visualization that looks like that!

Programming part

Task

The purpose of this assignment is to build a MinHeap in BRIDGES represented as a binary tree (as opposed to the more common array representation of a heap)

Recall that as a binary tree, a heap defined recursively as a root and two subheaps. The invariant of a min heap is that the root of any heap should have a lower (or equal) key than any node contained in the heap.

Getting Started

1. Open your scaffolded code.
2. Plug in your credentials.
3. Observe the `MyHeapElement` class that extends BRIDGES's `BinTreeElement`.
4. Observe the `MyHeap` class that provide Min Heap features.

Build a Binary Min Heap

1. Write the insert function in MyHeap. There are todos to guide you.

The algorithm for inserting in a heap is as follows. (This algorithm ignores that there is a key and a value.) Note that it uses information about the size of the subheaps being stored at each node of the heap.

```
Heap {
  Key
  HeapLeft
  SizeHeapLeft
  HeapRight
  SizeHeapRight
}

insert (Heap h, k) {
  if (h is empty)
    return makenewheap (k)

  if (k < h.Key)
    swap k and h.Key

  if (SizeHeapLeft < SizeHeapRight)
    //push left
    SizeHeapLeft = SizeHeapLeft + 1
    HeapLeft = insert (h.HeapLeft, k)
  else
    //push right
    SizeHeapRight = SizeHeapRight + 1
    HeapRight = insert (h.HeapRight, k)

  return h
}
```

If you have time

1. Use all of Shakespeare's works.
2. Change the location of the insert and pop in the main function to keep only the 100 most occuring words in the heap at any time.
3. Style the heap so that words with more than 1000 occurences are highlighted.

Help

for Java

Color documentation

BinTreeElement documentation

Shakespeare documentation

Bridges class documentation

for C++

Color documentation

BinTreeElement documentation

DataSource documentation

Shakespeare documentation