Program #4 Helper

Prof Bill - Apr 2020

For everything P4, see the Programs page.

thanks...yow, bill

You know this...

Quality Code™

Our CSC 210 coding guidelines are
wtkrieger.faculty.noctrl.edu/csc210-spring2020/docs/java_coding_guidelines.pdf

Repo with P4 shared code: github.com/williamt777/csc210-program4

Wed Apr 15

I uploaded V0.2 of our P4 shared code up to Github, github.com/williamt777/csc210-program4.

The graphs folder

You'll find a graphs folder on Github now. I created some small test files (testN.txt) and copied some of Sedgewick's examples too. The README describes the scene, man.

My Prim's results are in the graphs folder too. I saved my Prim's min spanning tree as a new, separate graph. I recommend you do likewise. The verts are the same as the original graph, but there at (V-1) edges.

And hey, **email me** when you get some Prim's results of your own. I'd like to see if our answers match. But how can we tell if our Prim's answers match?!? Keep reading...

Normalized (?!?)

I added methods to "normalize" an edge and a graph in our shared interfaces.

- edge.normalize() - make sure vert1 is smaller than vert2
- graph.normalizeEdges() - normalize each edge, and then sort all edges

After two graphs are normalized, I can check each edge in order to see if they are exactly equal: vert1, vert2, weight. If so, then the graphs are equal.

You can implement code to compare graphs. It's pretty fun and easy. Then, check your answers against mine. Or, contact me and I'll compare our Prim's graphs.

I think that's it.

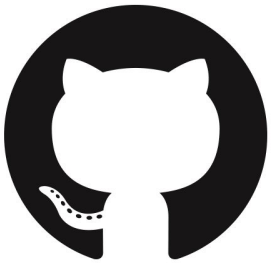
Your git/Github workflow

Reminder: Once your git folder and Github repo are setup, then your workflow will be:

```
# edit your files
$ git add -A
$ git commit -m "comment about your edits"
$ git push origin master
```

thanks...yow, bill

PS - the Github octocat logo, github.com/logos



Mon Apr 13

Suggestion: Create a **data folder** for the graph examples you want to run. Then, add "data" to your .gitignore file, so they aren't pushed up to your Github repo.

P4 has two steps: 1) your Graph210 implementation, and then 2) doing Prim's.

Step 1 - MyGraph: Some notes about your MyGraph implements Graph210:

- I keep an ArrayList of all edges because it's easy.
- I also have an adjacency list for each vertex. Two options here:
    - Do a list of lists or an array of lists. Java is cranky about doing this with generic data types like ArrrayList.
    - Create a class called AdjacencyList which holds the vertex and an ArrayList of edges connected to it. It's a little "bigger" code, but easier to put together and understand. I did this. I simply have an array of AdjacencyList objects, one per vert.
- On any error when reading a graph file (like a bad vert number), just print an error and give up (return null). No fancy error-handling is necessary.
- How to test your MyGraph? Read a graph. Then print it back out again.
- I started with this teeny-tiny example: 2 verts, 1 edge. Then, I graduated to Sedgewick's example: tinyEWG.txt with 8 verts.

```
2
1
0 1 1.0
```

Step 2: Prim's stuff:

- Here's some help doing a PriorityQueue with a Comparator, www.geeksforgeeks.org/implement-priorityqueue-comparator-java

- What to put in your PQ? I created a new class (PrimsRecord) to hold all my Prim's info: distance, parent, known. Now, it's easy to write a comparator.
- Prim's output? How about a new graph. Then just print using MyGraph.toString().
- I'll post my Prim's results later this week. The solution for Sedgewick's tinyEWG is at his site and below.

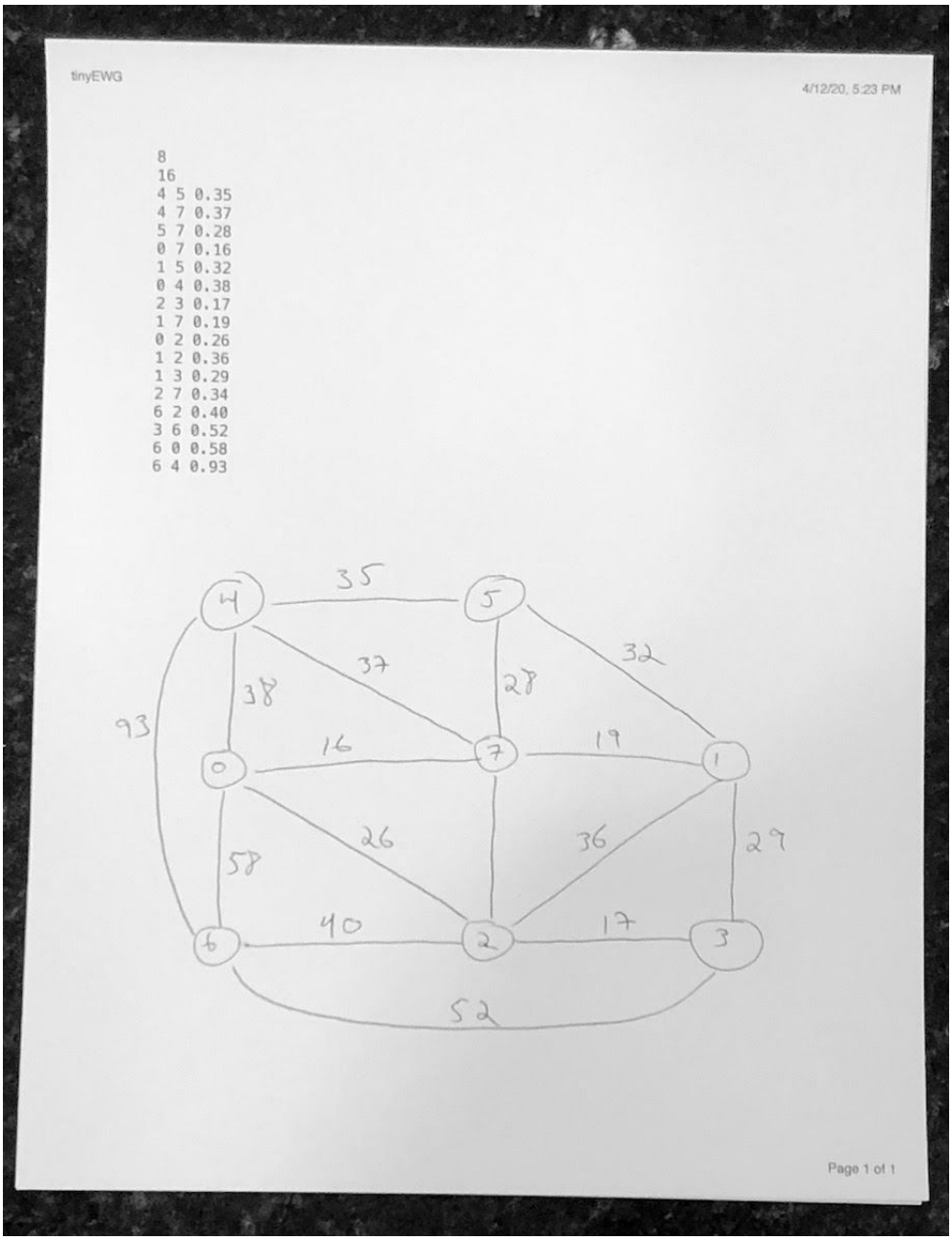
If you want to do Kruskal's let me know. It is very cool. I have some disjoint set code I can share with you.

thanks...yow, bill

Easter Sunday Apr 12

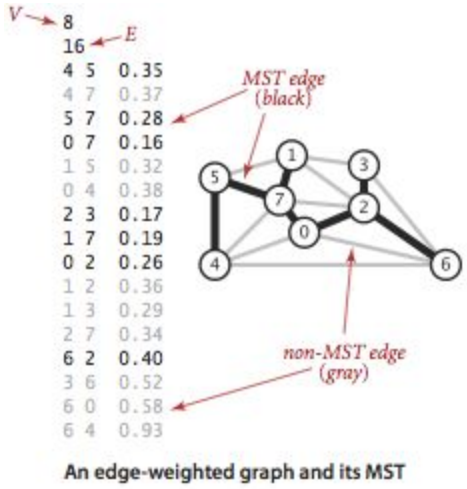
Sedgewick has some tasty example graphs, algs4.cs.princeton.edu/43mst/.

I sketched one out his smallest one, tinyEWG.txt. How's that?



Sedgewick gives the min spanning tree solution to tinyEWG.

Let's verify this by hand. Start at vert 0, of course.



An edge-weighted graph and its MST

Source: algs4.cs.princeton.edu/43mst/

Question: For a graph with V verts, how many edges does the min spanning tree have?

Wed Apr 8

The Graph210 design team has released V0.2 of our interfaces and updated our repo.

Remember our git/Github workflow!

```
# edit, edit, edit
$ git add -A
$ git commit -m "V0.2 of our interface"
$ git push origin master
```

The P4 shared code repo: github.com/williamt777/csc210-program4.

Use the **green** clone or download button.

We tested our interface by writing pseudocode for three examples: readfile, writeFile, and Prim's. Go!

```
// read a graph file, format is from Sedgwick
// Source: algs4.cs.princeton.edu/43mst
readFile( file)
  numVerts = read num
  numEdges = read num
  MyGraph g = new MyGraph( numVerts, numEdges)
  for i = 1 to numEdges
    if EOF, then print error return null
    vert1, vert2, weight = read 3 nums
    g.addEdge( vert1, vert2, weight) // error-check!
  end for
end readFile

// write a graph file, format is Sedgewick (again)
writeFile( Graph210 g)
  List<Edge210> e = g.allEdges();
  write g.numVerts()
  write g.NumEdges()
  e = g.allEdges()
  for each edge in e
    write vert1, vert2, weight
```

```
  end for
end writeFile

// find the min spanning tree using Prim's
// starting point was the Graph algorithms notes from class
doPrims( MyGraph g, startVert)
  create arrays: D, parent, known
  create pq = new PriorityQueue() // JCF

  D[startVert] = 0
  pq.add( startVert)
  while( ! pq.empty)
    u = pq.removeMin()
    known[u] = true
    edges = g.edges(u)
    for each edge e in edges
      v2 = e.otherVert( u)
      if ! known[v2] && e.getWeight() < D[v2]
        D[v2] = e.getWeight()
        parent[v2] = u
        update v2 in pq PriorityQueue // little work
      end if
    end for
  end while
end doPrims
```

Extra notes:

- Your **MyGraph implemements Graph210**. Use the adjacency list representation. So, you'll need 1) an array of edges, ArrayList<Edge210> and 2) an ArrayList of Edges for each vert.
- Sedgewick has 5 great graph files for us to use. Search on "tinyEWG", algs4.cs.princeton.edu/43mst/
- In Prim's you can use the **PriorityQueue** in the JCF. Google: java priorityqueue. Just fyi: The removeMin() method in PriorityQueue is called poll().

TODO: How do we write the min spanning tree? Write to a file?

thanks...yow, bill

PS - I'll be coding Good Friday at 1:20. *Optional!*

Mon Apr 6

Y'all now have a public face in nerd-land. Congrats!

Let's take a peek as of Monday morning and polish things up a bit.

- Emmanuel B, github.com/Eborishade, move your README.txt to README.md; move files to src and rm the original folder, use packages if you want "folders"
- Jason E, github.com/jsengland, mv README.txt to README.md
- Luis G, github.com/Lgonzalez36, rm test repo; mv README.txt to README.md; put lib folder and .vscode in .gitgnore; src files in current folder, rm ProjectHW10
- Vinny K, github.com/vkast00, beauty README!; rm bin folder, put in .gitignore, only src in Github
- Joe K, github.com/JosephKl, very nice!
- Cole P, github.com/CPaulline, mv src to current folder, rm Program03 folder; rm lib folder too!
- Brennan S, github.com/bssweeney, rename README.md.txt; look at Program03.java...beauty!
- Devon T, github.com/DCTomblinson/, rm test2.txt, then push
- John Z, coming soon…
- And ye olde Prof Bill, github.com/williamt777, organizations in bottom left

Sun Apr 5

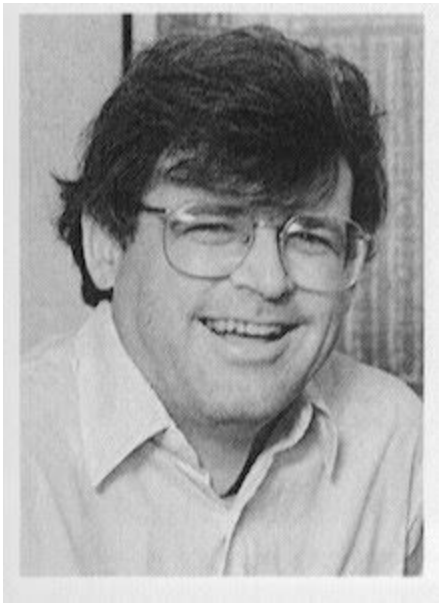
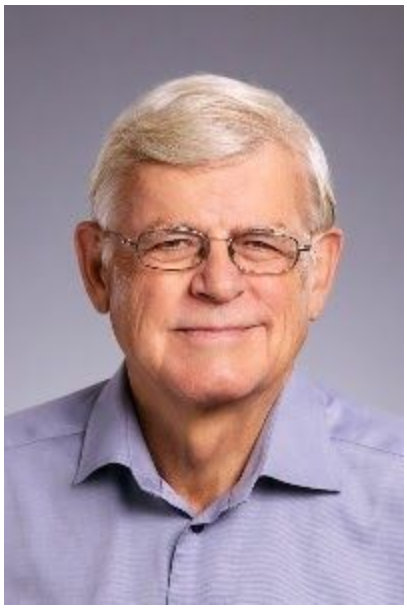
It's easy as one, two, three…

1. We'll work on the graph representation (Java interfaces) in class
2. Then, I'll publish our Java interfaces on Github
3. You coder 'er up

I lurve Sedgewick here. His "edge-weighted" graph. Notice the awesome examples too!
algs4.cs.princeton.edu/43mst

Idea: Maybe we can do a crowd210-sourced example too. If we all pitch-in 20 data points, then…We'll see.
thanks...yow, bill

PS - I give major kudos to Sedgwick at Princeton. His online text is excellent. But that kind of extra effort takes its toll over the years. Ha!

	
Sedgwick before	Sedgwick after