PART 3 readme

Things to note

I named my branch forreview before reviewing it and I realized renaming it back to refactor would remove the pull request so I left that as my working branch name. Also I didn't see anywhere saying we should merge the request back to the main branch so I left it as a request so it can be merged. There are no conflicts so I could merge it whenever.

Refactors

https://github.com/ryannav/CSE-464-2023-mrnavar2/commit/7e19138ed055c5ed78b0e1450dca77e977f98baa

Method extraction checknodes, this simplifies the way that the nodes are checked in multiple different methods including removeNode, removeNodes and addNodes and I believe others. This is to simplify the code and increase readability.

https://github.com/ryannav/CSE-464-2023-mrnavar2/commit/a74af15d85e340d968668b957ccfb823f83dabea

Renamed variables inside of search method to a name that fit better. Make the code much easier to understand since the name means something, it didn't mean anything before.

https://github.com/ryannav/CSE-464-2023-mrnavar2/commit/a0944963c546d8c0eafedc313c6e3228ab94359d

Extracted a chunk of identical code into a method in two spots and made it into a method that was called twice. This method is called outputStr. This also shortens the code and makes it simpler and easier to understand. It basically removed 2 sections of unnecessary code.

https://github.com/ryannav/CSE-464-2023-mrnavar2/commit/e487d92116403036265ea6b8223b4137bdeff950

fourth refactor, simplified arguments for the code in graphics shortening it and removing unnecessary code.

https://github.com/ryannav/CSE-464-2023-mrnavar2/commit/dfde2b67085461fe2e756e0420a7dd0530506cd1

fifth refactor, extracted the method that checks and prints if the edge doesn't already exist in the graph. This method is used in the code and could be used for future updates in the code so I found it would be helpful to extract it.

Template Design

https://github.com/ryannav/CSE-464-2023-mrnavar2/commit/eedd5648126ff31505bc7411a68f7ecb48434494

Created the classes BFSearch, DFSearch and SeachTemplate (Later also RandSearch) I used SearchTemplate as an abstract class to abstract all of the methods that would be necessary for the search methods. I also implemented ones that I could that would be the same among all of them. I then made each search class extend the template class and implemented all of the methods based off of the abstracted template class.

Strategy design

https://github.com/ryannav/CSE-464-2023-mrnavar2/commit/571241f89ed48d9ac390675e9bc1a4c3e42b5ac0

I created the SearchStrategy, SearchContext classes searchStrategy is the interface class which will give the behavior of the search strategies which is to search. Then I implemented the SearchContext which will set the behavior for the search based on how the strategy is called. I then added an implements to each of the search classes, since before I used the enum to call a method in the path class for each search I left it that way and had the enum call the method that

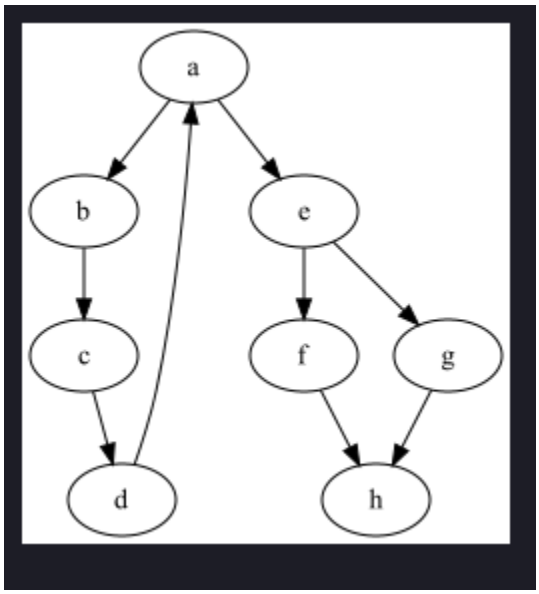
BFS AND DFS tests after refactors and design

Random walk

https://github.com/ryannav/CSE-464-2023-mrnavar2/commit/0d1118026a044060113210bb59712bf16052d2ff

Had it parse the graph and choose a random node to walk for each step into the graph. Used a random number to pick which adjacent node it would walk to in order to parse the rest of the graph until it reached the destination node.

Tests




```
ryannav *
@Test
public void testrand(){
    main s = new main();
    s.parseGraph( filepath: "/test3.dot");
    s.outputGraphics( path: "outputs/randwalktest.png", format: "png");
    String walk = s.randomWalkSearch(s.getNode( label: "a"), s.getNode( label: "h")).toString();
    assertEquals(walk.isBlank(), actual: false);    //checks that we get a path
}

new *
@Test
public void testrandextra(){
```

```
Tests passed: 1 of 1 test – 2 sec 60 ms
E:\JDK\bin\java.exe ...
a->e->g->h

Process finished with exit code 0
```

```
ryannav *
@Test
public void testrand(){
    main s = new main();
    s.parseGraph( filepath: "/test3.dot");
    s.outputGraphics( path: "outputs/randwalktest.png", format: "png");
    String walk = s.randomWalkSearch(s.getNode( label: "a"), s.getNode( label: "h")).toString();
    assertEquals(walk.isBlank(), actual: false);    //checks that we get a path
}

new *
@Test
public void testrandextra(){
```

```
Tests passed: 1 of 1 test – 2 sec 67 ms

E:\JDK\bin\java.exe ...
a->b->c->d->a->e->g->h

Process finished with exit code 0
```

```
ryannav *
@Test
public void testrand(){
    main s = new main();
    s.parseGraph( filepath: "/test3.dot");
    s.outputGraphics( path: "outputs/randwalktest.png", format: "png");
    String walk = s.randomWalkSearch(s.getNode( label: "a"), s.getNode( label: "h")).toString();
    assertEquals(walk.isBlank(), actual: false);    //checks that we get a path
}

new *
@Test
public void testrandextra(){
```

```
Tests passed: 1 of 1 test – 2 sec 126 ms

E:\JDK\bin\java.exe ...
a->b->c->d->a->b->c->d->a->e->f->h

Process finished with exit code 0
```

CODE REVIEW

https://github.com/ryannav/CSE-464-2023-mrnavar2/pull/5#issuecomment-1827156890

https://github.com/ryannav/CSE-464-2023-mrnavar2/pull/5