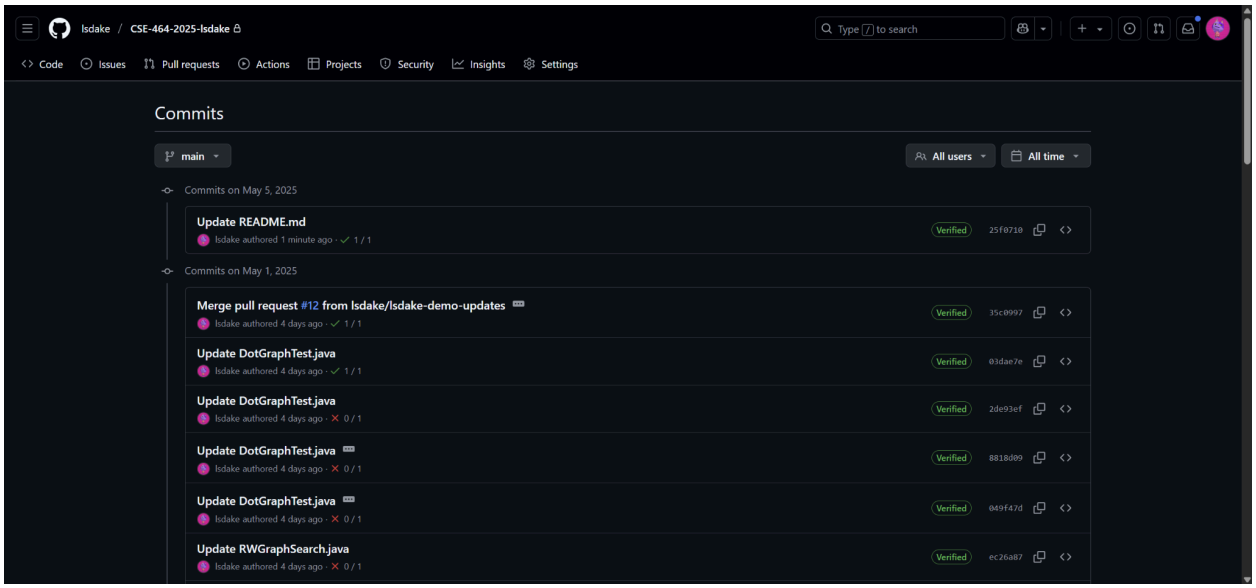
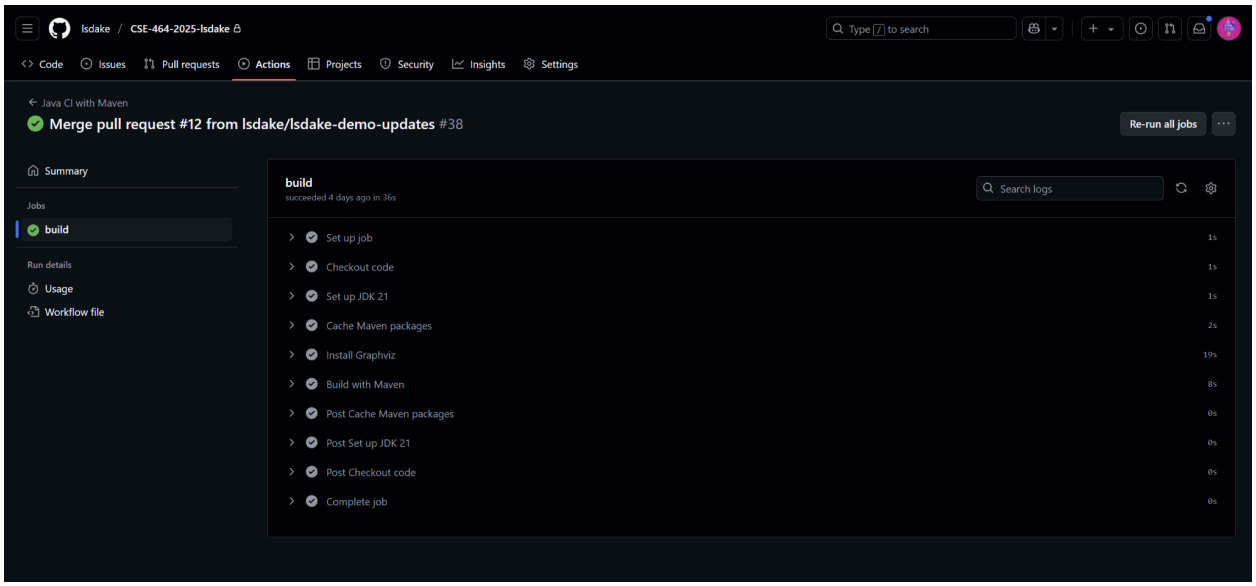
Go to my main README.md in the program for more in-depth info!

https://github.com/lsdake/CSE-464-2025-lsdake
Github Commits:



Continuous Integration Working:



Search Algorithms:

- Given a graph.dot file as follows:

```
digraph G {
   A -> B;
   A -> C;
   B -> D;
   C -> E;
   D -> F;
   E -> F;
}
```

- BFS
    - System.out.println(graph.GraphSearch("A", "F", graph.getBFS()).toString()); should output
    - A -> B -> D -> F

- DFS
    - System.out.println(graph.GraphSearch("A", "F", graph.getDFS()).toString()); should output
    - A -> C -> E -> F

- Random Search Algorithms:

- Given a graph.dot file as follows:

```
digraph G {
   A -> B;
   B -> C;
   C -> D;
   D -> A;
   A -> E;
   E -> F;
   G -> C;
   E -> G;
   F -> H;
   G -> H;
   H -> D;
}
```

- Fully Random Search
    - Given startNode == "A" and endNode == "H":

    - System.out.println(graph.GraphSearch(startNode, endNode, Algorithm.BFS).getPathArray());

  should output one of the following:
    - a->e->g->h (Target node!)
    - a->e->f->h (Target node!)
    - a->b->c->d (Dead end)
    - a->e->g->c->d (Dead end)

- Unvisited Random Search
    - Given startNode == "A" and endNode == "H":

    - System.out.println(graph.GraphSearch(startNode, endNode, Algorithm.BFS).getPathArray());

  should output one of the following statements:
    - a->e->f->h (Target node!)
    - a->e->g->h (Target node!)
    - a->b->c->d (Dead end)

- Unvisited Random Search
    - Given startNode == "A" and endNode == "H":

    - System.out.println(graph.GraphSearch(startNode, endNode, Algorithm.BFS).getPathArray());

  should output one of the following statements:
    - a->e->f->h (Target node!)
    - a->e->g->h (Target node!)
    - a->e->g->c->d (Dead end)
    - a->b->c->d (Dead end)