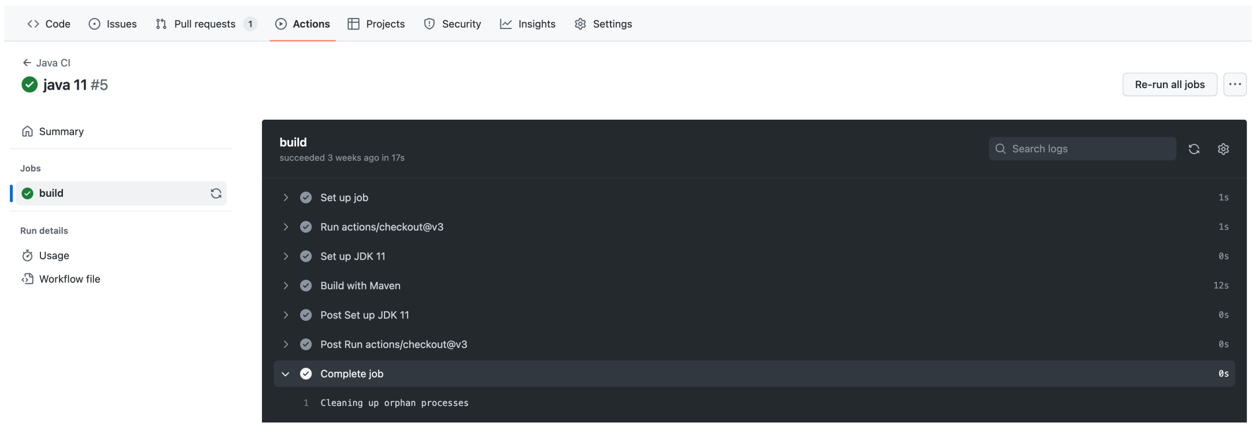
CSE 464: Software QA and Testing
Project Part #2

1. Add 3 new APIs and their test cases to support removing nodes and edges
   - Remove a node API (5 points): removeNode(String label)
   - Remove nodes API (5 points): removeNodes(String[] label)
   - Remove an edge API (10 points): removeEdge(String srcLabel, String dstLabel)
   - Add test cases for these 3 APIs. The test cases should cover at least the following scenarios:
     - Scenario 1: some nodes and some edges are correctly removed.
     - Scenario 2: removing nodes that do not exist in the graph will cause exceptions.
     - Scenario 3: removing edges that do not exist in the graph will cause exceptions.

2. Add continuous integration support for your github project: https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration. The TA/Graders should be able to see that once new code is checked in, the workflow of github will automatically build and test your code (20 points).
   - Below is an example screenshot for successfully executed workflow in github:



```
build
succeeded 3 weeks ago in 17s

> Set up job                      1s
> Run actions/checkout@v3         1s
> Set up JDK 11                   0s
> Build with Maven               12s
> Post Set up JDK 11              0s
> Post Run actions/checkout@v3    0s
v Complete job                    0s
  1  Cleaning up orphan processes
```

3. Create a branch bfs (https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-and-deleting-branches-within-your-repository), and add a graph search API in your graph manager class to this branch (20 points):
   - API: Path GraphSearch(Node src, Node dst)
   - This API finds a possible path from a node src to another node dst in a graph if it exists; otherwise, it returns null.
   - Path is a new class that you define to represent a path: n1 -> n2 -> n3
   - Use BFS algorithm (https://en.wikipedia.org/wiki/Breadth-first_search) to search for the path
   - Push this feature as at least a commit to the github repo
4. Create another branch dfs, and implement the same graph search API using DFS algorithm (https://en.wikipedia.org/wiki/Depth-first_search) (20 points):
   - API: Path GraphSearch(Node src, Node dst)
   - This API finds a possible path from a node src to another node dst in a graph if it exists; otherwise, it returns null.

- Path is a new class that you define to represent a path: n1 -> n2 -> n3
- Use DFS algorithm to search for the path
- Push this feature as at least a commit to the github repo

5. Merge the bfs branch to the main branch, and then merge the dfs branch to the main branch. You should see merge conflict. Resolve the merge conflict by changing the search API to add an enum parameter for choosing search algorithm (20 points):
   - API: Path GraphSearch(Node src, Node dst, Algorithm algo)
   - This API finds a possible path from a node src to another node dst in a graph using a specified algorithm (either bfs or dfs) if it exists; otherwise, it returns null.
   - Path is a new class that you define to represent a path: n1 -> n2 -> n3
   - Java enum: https://www.w3schools.com/java/java_enums.asp

Write a readme (in PDF) that provides the instructions and example code to run your code. TA and graders will run your code using your provided example inputs, run your tests, and run your code using our own inputs. You should submit your code (including your used libraries) and the readme in a zip file. The submission must be accepted via Canvas.

The readme PDF must meet the following requirements, or up to 20 points will be deducted:

- The github link must be provided in the google sheet: https://docs.google.com/spreadsheets/d/1JNQMnj9Z5gl9vc6PIRtqmkYQCJNwlXVKBKOpi9C-e4s/edit?usp=sharing
- The readme PDF must be checked into the repo, or attach in the comments of your canvas submission.
- Your code must be able to be compiled by typing "mvn package".
- You should provide screenshots of the expected output for each feature in the readme PDF, so that the TA/graders can easily check your results after they run your code.
- You must provide links to the github commits for each feature and each branch. For example, you must provide a link for showing your continuous integration work properly in github, and you should provide links for your branches and successful merges.
- You don't have to check in your built binaries and other built artifacts.