CSE 464: Software QA and Testing
Project Part #3

1. Please create a new branch, named refactor, to refactor your project. Please perform 5 refactoring on your code base and push the changes to the github. For each refactoring, you must clearly explain what type of refactoring you are applying, why you apply this refactoring, and the changes should be pushed as one commit. You must provide links for each of your refactoring commit in your readme PDF (10 points).

2. In your refactor branch, apply design patterns to refactor the code for BFS and DFS. The first design pattern to be applied is template pattern. That is, you need to abstract the common steps of BFS and DFS, e.g., fetching next node to search, and apply template pattern to extract these common steps in a base class and implement the different algorithms in the sub classes, respectively. These changes should be pushed to the branch in github and the links of the commits must be provided in the readme PDF (30 points).
    - Template pattern: https://www.geeksforgeeks.org/template-method-design-pattern/
    - In your readme PDF, you must provide explanation on how you apply the template pattern to extract common behaviors of BFS and DFS, and how the classes are created to implement the differences of BFS and DFS.

3. In your refactor branch, continue to apply strategy pattern so that we can choose BFS or DFS based on the algo parameter in the API: Path GraphSearch(String src, String dst, Algorithm algo). These changes should be pushed to the branch in github and the links of the commits must be provided in the readme PDF (20 points).
    - Strategy pattern: https://www.geeksforgeeks.org/strategy-pattern-set-1/ and https://www.geeksforgeeks.org/strategy-pattern-set-2/?ref=lbp
    - In your readme PDF, you must provide explanation on how you apply the strategy pattern to choose whether to use BFS or DFS, and how the classes are created to implement the choices of BFS and DFS.

4. In your refactor branch, please add a new graph search algorithm, random walk search. This is a simplified version of random walk: given a node `src` as the beginning node of the search, the next node to search is a random node `k` where `k` is one of `src`'s neighbors that can be reached from `src` via an edge. That is, by applying this algorithm, you may find a path from a source node `src` to a destination node `dst` through random search of the destination nodes (20 points).
   - You must implement this new random walk algorithm using both the strategy pattern and the template pattern you implement for requirements 2 and 3 in this part.
   - You must provide an example graph and an example search to find a path in the example graph.
   - You must make your code can search a path from `a` to `c` using the provided graph file at canvas
   - You must run the search multiple times and output the search process to show the randomness of your search.
     - Example outputs:

```
random testing
visiting  Path{nodes=[Node{a}]}
visiting Path{nodes=[Node{a}, Node{b}]} visiting
Path{nodes=[Node{a}, Node{e}]} visiting
Path{nodes=[Node{a}, Node{e}, Node{g}]} visiting
Path{nodes=[Node{a}, Node{e}, Node{g}, Node{h}]}
visiting Path{nodes=[Node{a}, Node{b}, Node{c}]}
Path{nodes=[Node{a}, Node{b}, Node{c}]}
```

```
random testing
visiting  Path{nodes=[Node{a}]}
visiting  Path{nodes=[Node{a},  Node{e}]}  visiting
Path{nodes=[Node{a}, Node{e}, Node{g}]} visiting
Path{nodes=[Node{a}, Node{e}, Node{g}, Node{h}]}
visiting  Path{nodes=[Node{a}, Node{e}, Node{f}]}
visiting  Path{nodes=[Node{a},  Node{b}]}  visiting
Path{nodes=[Node{a}, Node{b}, Node{c}]}
Path{nodes=[Node{a}, Node{b}, Node{c}]}
```

```
random testing
visiting  Path{nodes=[Node{a}]}
visiting Path{nodes=[Node{a}, Node{b}]} visiting
Path{nodes=[Node{a}, Node{b}, Node{c}]}
Path{nodes=[Node{a}, Node{b}, Node{c}]}
```

5. We will conduct a code review of your changes with the TA/Graders. After you finish all three features, create a pull request, and then enter your pull request link at https://docs.google.com/spreadsheets/d/1YIXeYnzskgnpDyh9BlB5qit1tzw1nVGJMmu2J1EbXPk/edit?usp=sharing. Note that some students still do not provide their repo links. Not providing your repo link or your pull request link will result in a 5 point deduction of your course project grade. Once you have created the pull request, please email a student in this class with the pull request link by **April 18** and ask him/her to provide code reviews on your changes. Late submission of the link will result in 5 point deduction and may not receive code review. Please respond to all the reviews and make code changes to address the reviews (20 points).
   - Github pull request: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request
   - Pull request review: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/reviewing-changes-in-pull-requests/about-pull-request-reviews
   - Commit changes to pull request: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/committing-changes-to-a-pull-request-branch-created-from-a-fork

Write a new readme (in PDF) that provides the instructions and example code to run your code. Your PDF should also include screenshots of your commits and github workflow results to show that your continuous integration is setup correctly and your merge works as expected. 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.

Additional submission requirement:

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/1YIXeYnzskgnpDyh9BlB5qit1tzw1nVGJMmu2J1EbXPk/edit?usp=sharing.
- **Do not overwrite your previous readme PDFs. Please create a new readme PDF. Missing the readme file will cause 5 point deduction.**
- The new 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 must provide detailed instructions to run BFS/DFS/Random Search of your new features.
- 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. For example, you must provide a link for showing your refactoring 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.