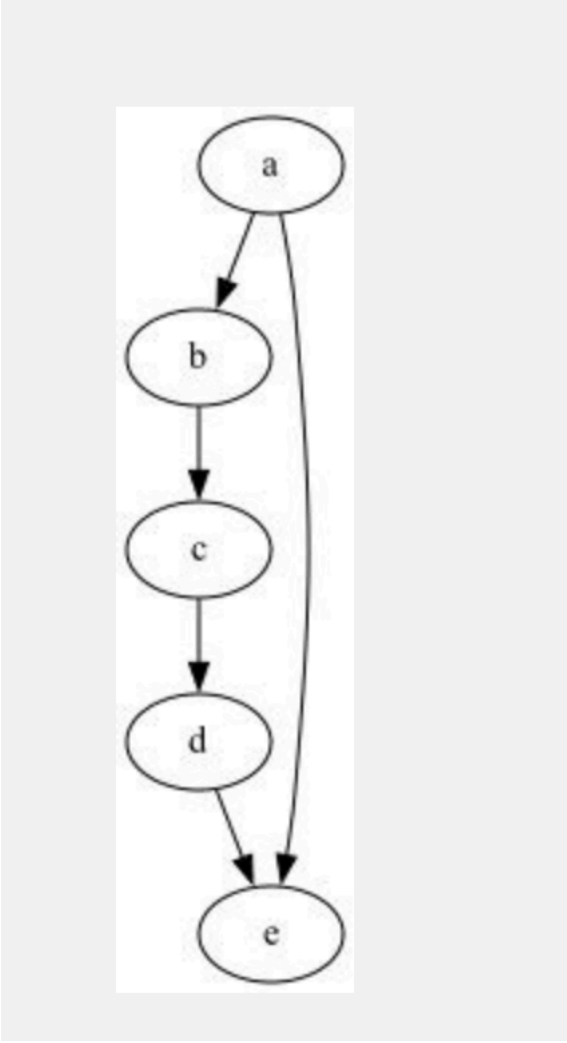
ReadMe

GithubLink: https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy

Explanation of BFS and DFS:

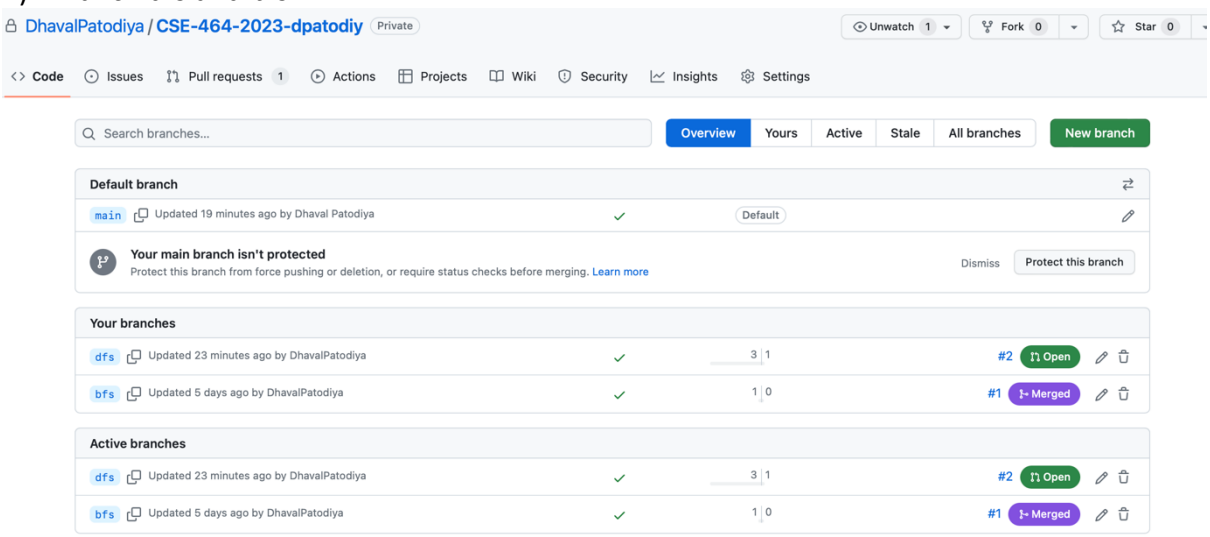
- Both the algorithm will not detect path if the node is not present and will return null as output.
- Both algorithm will not detect path if there is no path from Node A to Node B.
- Both algorithm will return path in form of object of “Path Class” and can be converted to string using pathobject.toString() (“a->b->c)
- Both algorithm will take ‘src’, ‘dst’ as String and ‘algo’ as Algorithm enum.
- If the pathSearch API is provided with invalid names of Algorithm then it will throw error.
- The graph will not find path form Node A to Node A unless and until there is a cycle from Node A to Node A. If there is no incoming edges in Node A then the path will not be found and will be returned as null.

Input Graph:

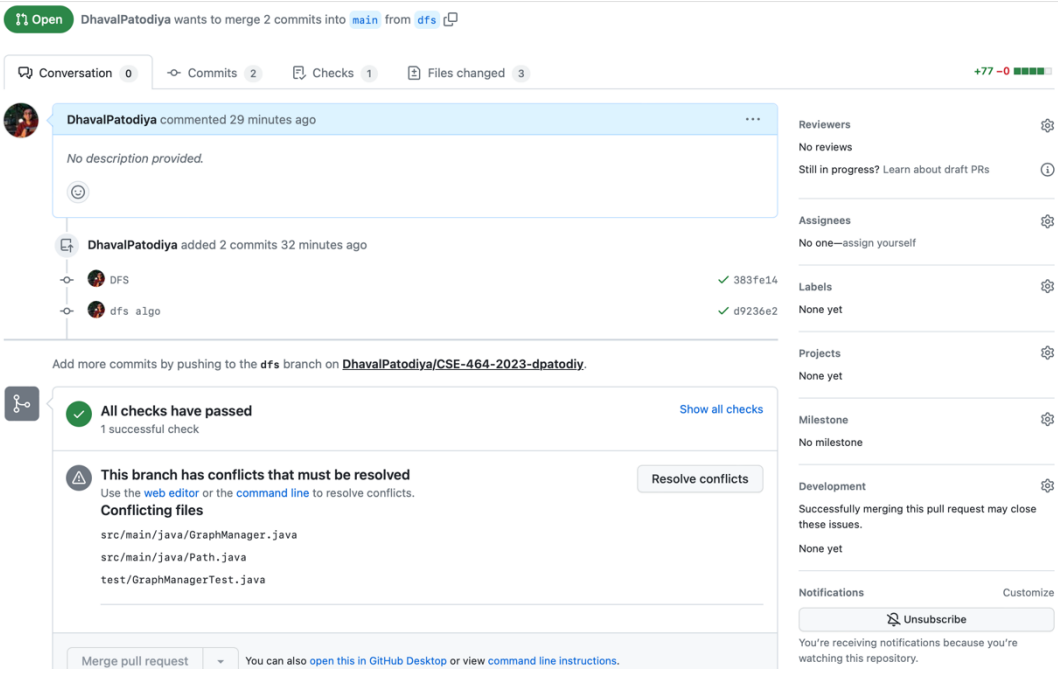


Branch And commits:

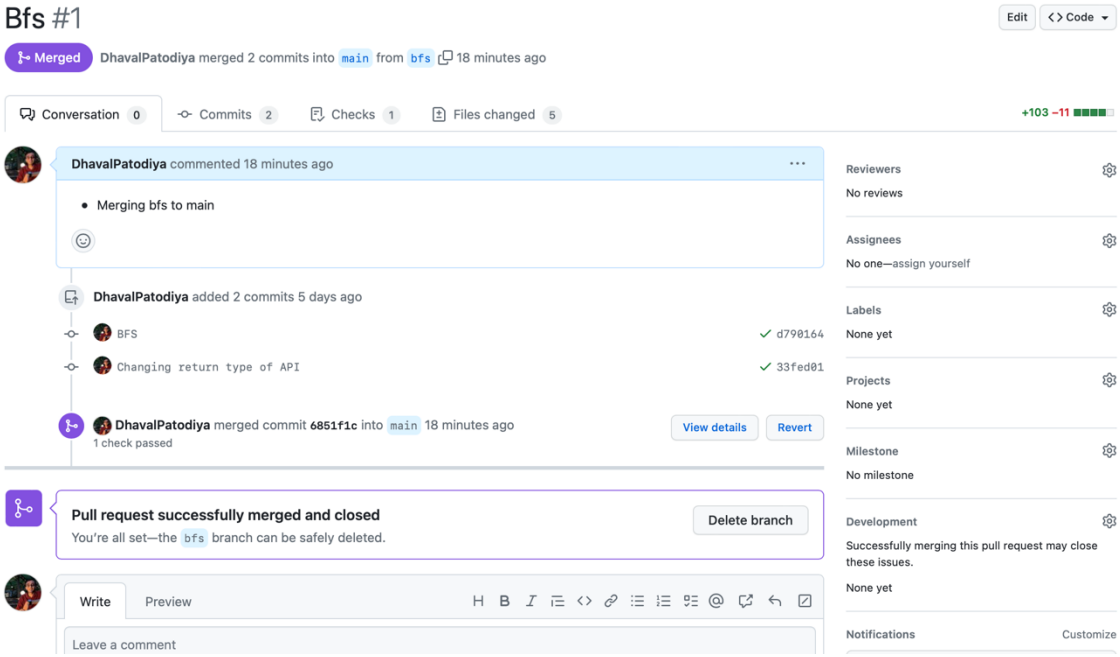
1) Branch dfs and bfs



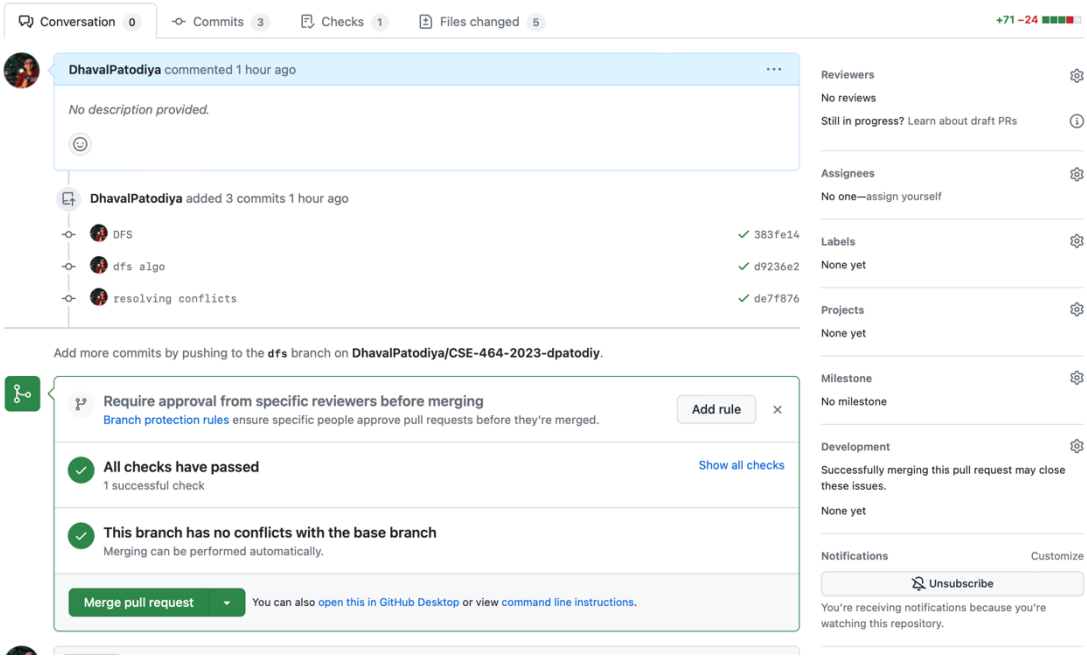
2) Merge conflict of branch dfs in main after merging main
   - DFS commit for merge conflict: https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/d9236e237d869801807f3d0f04bb010ce64f5103



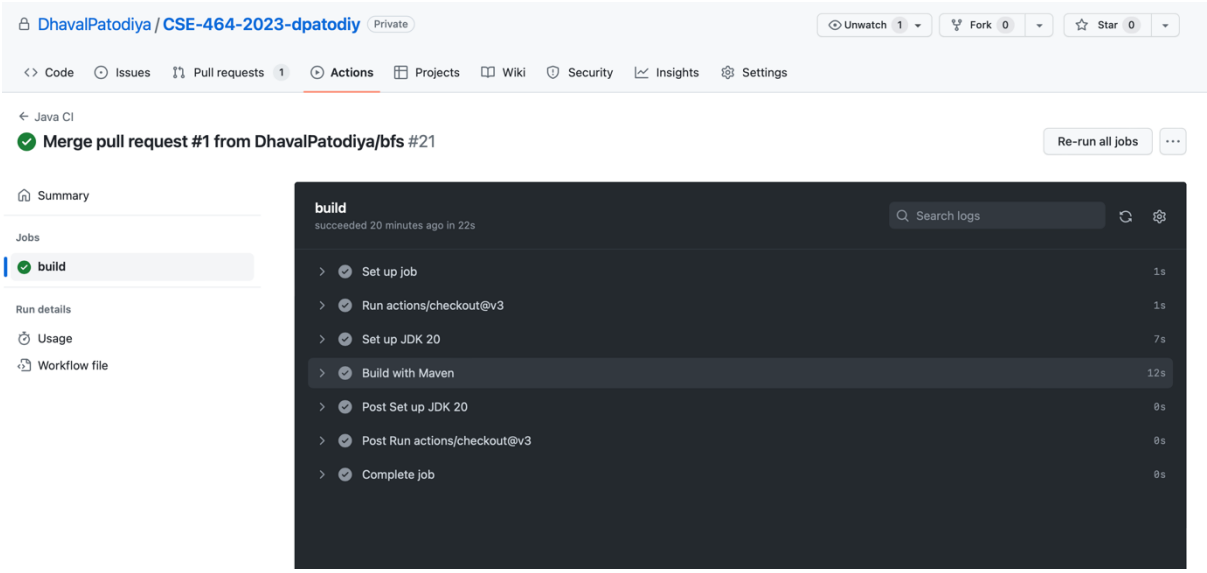
3) Merging BFS to Main before merging DFS.
    - BFS commit of merge: https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/33fed0162f675180bda398279f99597e392bc4fe



4) Github CI setup properly and successful runs:
    - Commit id of first successful run : https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/724819bdf576dc7b620c351d493587eaf0740e4f
    - First failed run : https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/d5714c3085fd200d805670916ce16c686817df48



5) Resolved Conflicts
   - Commit id: https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/pull/2/commits/de7f8761a6fe6e84d39e3eee3c944a44b3affac1
   - Successful merge id: https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/pull/2



Instructions and example code

I have written test cases providing the expected value and actual values explaining with comments. I am attaching the screenshot of test cases here providing detailed explanations:

1) BFS algo with path found:

```
// a-d poth exists
@Test
public void bfsPathFound() throws Exception{
    String expected = "a->b->c->d";
    Path actual = g.graphSearch( src: "a",  dst: "d", Algorithm.BFS);
    Assert.assertTrue(expected.equals(actual.toString()));
}
```

2) BFS algo with path not found from node itself to node itself: As once we leave the node we can't reach it meaning there is no cycle from a to a.

```
//from a to a there is no path like a->b->a, there is no path to return to a once you leave a
@Test
public void bfsPathNotFound() throws Exception{
    Path actual = g.graphSearch( src: "a",  dst: "a", Algorithm.BFS);
    Assert.assertNull(actual);
}
```

3) BFS algo When path doesn't exists as node f doesn't exists

```
// a-f path doesn't exists as f doesn't exists.
@Test
public void bfsPathNotFoundNode() throws Exception{
    Path actual = g.graphSearch( src: "a",  dst: "f", Algorithm.BFS);
    Assert.assertNull(actual);
}
```

4) DFS algo with path exists between src and dst.

```
//b-e path exists
@Test
public void dfs() throws Exception{
    String expected = "b->c->d->e";
    Path actual = g.graphSearch( src: "b",  dst: "e", Algorithm.DFS);
    Assert.assertEquals(expected, actual.toString());
}
```

5) DFS algo with path not found from node itself to node itself: As once we leave the node we can't reach it meaning there is no cycle from a to a.

```
//from a to a there is no path like a->b->a, there is no path to return to a once you leave a
@Test
public void dfsPathNotFound() throws Exception{
    Path actual = g.graphSearch( src: "a",  dst: "a", Algorithm.DFS);
    Assert.assertNull(actual);
}
```

6) DFS algo with node doesn't exits will return null. a-g path is not present as there is no node g.

```
// a-g path doesn't exists as g doesn't exists.
@Test
public void dfsPathNodeNotFound() throws Exception{
    Path actual = g.graphSearch( src: "a",  dst: "g", Algorithm.DFS);
    Assert.assertNull(actual);
}
```

7) IllegalArgumentException thrown when algo is other than DFS or BFS.

```
//IllegalArgumentException thrown when algo is other than BFS or DFS
@Test(expected= IllegalArgumentException.class)
public void invalidEnumName() throws Exception{
    System.out.println(g.graphSearch( src: "b",  dst: "e", Algorithm.valueOf("DMS")));
}
```

8) NullPointerException thrown when algo is null.

```
//NullPointerException thrown when algo is null
@Test(expected= NullPointerException.class)
public void algoNameIsNull() throws Exception{
    System.out.println(g.graphSearch( src: "b",  dst: "e",  algo: null));
}
```

Git Commit TimeLine