

QAMP Checkpoint 2

Work done after Checkpoint 1:

1. Practice how to write and form programs in Rust

Doing this was easy since I already had a good hand over C/C++. However, studying advanced syntax methods/techniques required at the level of the repository were a bit hard to catch up at first. It took me quite some time to get used to writing a program in Rust.

2. Construct program for implementing Triangular Lattice Graph (graph generator)

The overall code segment for implementing a graph generator is completed. Although there are still some errors and failing tests in the prepared code, I am working on rectifying them. The next step would be to push the code onto the repository and perform further tests to finally merge.

3. Understand basic GitHub commands and automated Python testing

Grasped a few new techniques and libraries that are, for example, used to automate Python testing (tox) and building Rust programs (cargo build). These commands are used quite frequently and overall helped me understand how open-source development is done.

4. Working on a few more graph generators

I am simultaneously trying to resolve issues with the triangular lattice program and also code other generators like a few random graphs.

Any difficulties faced?

Not much but it is sometimes hard to figure out why tests are failing and the rectification process consumes a lot of time.

Required Visual Representation:

1. Code building

```
File Edit Selection Find View Goto Tools Project Preferences Help
generators.rs x
2219
2220 #[pyfunction(multigraph = true)]
2221 #[pyo3(text_signature = "(rows, cols, /, multigraph=False)")]
2222 pub fn triangular_lattice_graph(
2223     py: Python,
2224     rows: usize,
2225     cols: usize,
2226     multigraph: bool,
2227 ) -> graph::PyGraph {
2228     let mut graph = StablePyGraph::::default();
2229
2230     if rows == 0 || cols == 0 {
2231         return graph::PyGraph {
2232             graph,
2233             node_removed: false,
2234             multigraph,
2235         };
2236     }
2237
2238     let mut rowlen = rows;
2239     let mut collen = cols;
2240     let mut diff;
2241
2242     // Nodes in a row
2243     let num_row_nodes = (collen + 1) / 2;
2244
2245     rowlen = rowlen + 1;
2246     collen = num_row_nodes + 1;
2247
2248     let mut rm_num = 0;
2249     if cols % 2 != 0 {
2250         for _j in (1..rowlen).step_by(2) {
2251             rm_num += 1;
2252         }
2253     }
2254
2255     let num_nodes = (collen * rowlen) - rm_num;
2256
2257     let nodes: Vec<NodeIndex> =
2258         (0..num_nodes).map(|_| graph.add_node(py.None())).collect();
2259
2260     // Horizontal edges - col
2261     // Excluding last row
2262     for j in 0..(rowlen - 1) {
2263         diff = j * collen;
2264         for i in 0..(collen - 1) {
2265             graph.add_edge(nodes[diff + i], nodes[diff + (i + 1)], py.None());
2266         }
2267     }
2267
Line 1, Column 1
```

```
File Edit Selection Find View Goto Tools Project Preferences Help
generators.rs x
2285
2286     if rm_num == 1 {
2287         diff = (rowlen - 2) * collen;
2288         for i in (0..collen).step_by(2) {
2289             if i == 0 {
2290                 graph.add_edge(nodes[diff + i], nodes[diff + collen], py.None());
2291             }
2292             else {
2293                 for c in 1..(collen - rm_num) {
2294                     graph.add_edge(nodes[diff + i], nodes[diff + collen + c], py.None());
2295                 }
2296             }
2297         }
2298     } else {
2299         diff = (rowlen - 2) * collen;
2300         for i in 0..collen {
2301             graph.add_edge(nodes[diff + i], nodes[diff + (i + 1) + collen - 1], py.None());
2302         }
2303     };
2304
2305     // Diagonals except last row
2306     for j in 0..(rowlen - 2) {
2307         diff = j * collen;
2308         for i in 1..(collen - 1) {
2309             graph.add_edge(nodes[diff + i], nodes[diff + (i + 1) + collen], py.None());
2310             graph.add_edge(nodes[diff + i], nodes[diff + (i + 1) + collen - 2], py.None());
2311         }
2312     }
2313
2314     // last row
2315     if rm_num == 1 {
2316         diff = (rowlen - 2) * collen;
2317         for i in 1..(collen - rm_num) {
2318             graph.add_edge(nodes[diff + i], nodes[diff + collen], py.None());
2319             graph.add_edge(nodes[diff + i], nodes[diff + collen + 1], py.None());
2320         }
2321     } else {
2322         diff = (rowlen - 2) * collen;
2323         for i in 0..collen {
2324             graph.add_edge(nodes[diff + i], nodes[diff + (i + 1) + collen - 1], py.None());
2325         }
2326     }
2327
2328     graph::PyGraph {
2329         graph,
2330         node_removed: false, // or true?
2331         multigraph,
2332     }
2333
Line 1, Column 1
```

2. Creating Tests for the above code

```
File Edit Selection Find View Goto Tools Project Preferences Help
test_triangular_lattice.py x
1 import unittest
2
3 import networkx
4 from networkx.visualization import mpl_draw
5 import matplotlib.pyplot as plt
6
7 class TestTriangularLatticeGraph(unittest.TestCase):
8
9     def test_triangular_lattice_graph_edge_2_3(self):
10         graph = networkx.generators.triangular_lattice_graph(2, 3)
11         edge_list = graph.edge_list()
12         expected_edge_list = [
13             (0, 1),
14             (1, 2),
15             (3, 4),
16             (4, 5),
17             (0, 3),
18             (1, 4),
19             (2, 5),
20             (3, 6),
21             (5, 7),
22             (1, 3),
23             (1, 5),
24             (4, 6),
25             (4, 7), ]
26
27         self.assertEqual(edge_list, expected_edge_list)
28         # self.assertEqual(len(graph), 16)
29         # self.assertEqual(len(graph.edges()), len(expected_edges))
30         # self.assertEqual(list(graph.edge_list()), expected_edges)
31         # mpl_draw(graph)
32         # plt.show()
33
34     def test_triangular_lattice_graph_edge_3_4(self):
35         graph = networkx.generators.triangular_lattice_graph(3, 4)
36         edge_list = graph.edge_list()
37
38         expected_edge_list = [
39             (0, 1),
40             (1, 2),
41             (3, 4),
42             (4, 5),
43             (6, 7),
44             (7, 8),
45             (9, 10),
46             (10, 11),
47             (0, 3),
48             (1, 4),
49             (2, 5)]

```

Line 1, Column 1: Detect Indentation: Setting indentation to 4 spaces

3. Building the Rust code - using cargo build

```
cd Sub
cd rus
(base) saasha@Inspiron:~$ cd SublimeText/
(base) saasha@Inspiron:~/SublimeText$ cd rust/
(base) saasha@Inspiron:~/SublimeText/rust$ cd networkx
(base) saasha@Inspiron:~/SublimeText/rust/networkx$ cd src/
(base) saasha@Inspiron:~/SublimeText/rust/networkx/src$ subl generators.rs
(base) saasha@Inspiron:~/SublimeText/rust/networkx/src$ conda activate networkx-env
(networkx-env) saasha@Inspiron:~/SublimeText/rust/networkx/src$ cargo build
Finished dev [unoptimized + debuginfo] target(s) in 1.74s
```

4. Running the tests - using tox -epy

```
(networkx-env) saasha@Inspiron:~/SublimeText/rust/networkx/src$ tox -epy
GLOB sdist-make: /home/saasha/SublimeText/rust/networkx/setup.py
py inst-nodesps: /home/saasha/SublimeText/rust/networkx/.tox/.tmp/package/1/networkx-0.11.0.zip
py installed: attrs==21.2.0,autopage==0.4.0,cliff==3.9.0,cmd2==2.2.0,colorama==0.4.4,cycler==0.11.0,extras==1.0.0,fixtures==3.0.0
pbrr==5.7.0,Pillow==8.4.0,prettytable==2.4.0,pyparsing==3.0.4,pyperclip==1.8.2,python-dateutil==2.8.2,python-subunit==1.4.0,PyYAML
,steestr==3.2.1,stevedore==3.5.0,testtools==2.5.0,toml==0.10.2,voluptuous==0.12.2,wcwidth==0.2.5
py run-test-pre: PYTHONHASHSEED='2805345702'
py run-test: commands[0] | steestr run
[3] digraph.test_adj.TestAdj.test_out_direction [0.000467s] ... ok
[3] digraph.test_adjacency_matrix.TestDAGAdjacencyMatrix.test_default_weight [0.000524s] ... ok
[3] digraph.test_adjacency_matrix.TestDAGAdjacencyMatrix.test_negative_weight [0.000240s] ... ok
[3] digraph.test_adjacency_matrix.TestDAGAdjacencyMatrix.test_no_weight_fn [0.000594s] ... ok
[3] digraph.test_adjacency_matrix.TestDAGAdjacencyMatrix.test_random_graph_full_path [0.003761s] ... ok
[3] digraph.test_adjacency_matrix.TestFromComplexAdjacencyMatrix.test_random_graph_complex_dtype [0.000575s] ... ok
[3] digraph.test_all_simple_paths.TestDAGAllSimplePaths.test_all_simple_path_no_path [0.000302s] ... ok
[3] digraph.test_all_simple_paths.TestDAGAllSimplePaths.test_all_simple_paths_min_depth [0.000118s] ... ok
[3] digraph.test_all_simple_paths.TestDAGAllSimplePaths.test_graph_digraph_all_simple_paths [0.000390s] ... ok
[3] digraph.test_ancestors_descendants.TestAncestors.test_ancestors [0.000308s] ... ok
[3] digraph.test_ancestors_descendants.TestDescendants.test_descendants_no_ancestors [0.000296s] ... ok
[3] digraph.test_astar.TestAstarDigraph.test_astar_null_heuristic [0.000306s] ... ok
[3] digraph.test_avg_shortest_path.TestAvgShortestPath.test_simple_example [0.000479s] ... ok
[3] digraph.test_avg_shortest_path.TestAvgShortestPathAsUndirected.test_disconnected_graph [0.000438s] ... ok
[3] digraph.test_avg_shortest_path.TestAvgShortestPathAsUndirected.test_parallel_grid [0.021085s] ... ok
[3] digraph.test_avg_shortest_path.TestAvgShortestPathAsUndirected.test_partially_connected_graph [0.000348s] ... ok
[3] digraph.test_centrality.TestCentralityDiGraph.test_betweenness_centrality [0.000106s] ... ok
[3] digraph.test_centrality.TestCentralityDiGraph.test_betweenness_centrality_endpoints [0.000089s] ... ok
[3] digraph.test_centrality.TestCentralityDiGraph.test_betweenness_centrality_parallel [0.000152s] ... ok
[3] digraph.test_centrality.TestCentralityDiGraph.test_betweenness_centrality_unnormalized [0.000084s] ... ok
[3] digraph.test_centrality.TestCentralityDiGraphDeletedNode.test_betweenness_centrality [0.000090s] ... ok
[3] digraph.test_centrality.TestCentralityDiGraphDeletedNode.test_betweenness_centrality_endpoints [0.000078s] ... ok
[3] digraph.test_collect_bicolor_runs.TestCollectBicolorRuns.test_color_with_ignored_edge [0.000092s] ... ok
[3] digraph.test_collect_bicolor_runs.TestCollectBicolorRuns.test_filter_function_inner_exception [0.000096s] ... ok
[3] digraph.test_collect_runs.TestCollectRuns.test_cx_h_cx [0.000098s] ... ok
[3] digraph.test_collect_runs.TestCollectRuns.test_empty [0.000068s] ... ok
[3] digraph.test_collect_runs.TestCollectRuns.test_multiple_successor_edges [0.000079s] ... ok
[3] digraph.test_complement.TestComplement.test_empty_directed [0.000128s] ... ok
[3] digraph.test_compose.TestCompose.test_edge_map_and_node_map_funcs_digraph_compose [0.000126s] ... ok
[3] digraph.test_core_number.TestCoreNumber.test_directed_all_3 [0.000102s] ... ok
[3] digraph.test_deepcopy.TestDeepcopy.test_deepcopy_with_holes [0.000127s] ... ok
[3] digraph.test_deepcopy.TestDeepcopy.test_isomorphic_compare_nodes_identical [0.000182s] ... ok
```

Captured traceback:

Traceback (most recent call last):

File "/home/saasha/SublimeText/rust/networkx/tests/generators/test_triangular_lattice.py", line 27, in test_triangular_lattice
self.assertEqual(edge_list, expected_edge_list)

File "/usr/lib/python3.8/unittest/case.py", line 912, in assertEquals
assertion_func(first, second, msg=msg)

File "/usr/lib/python3.8/unittest/case.py", line 905, in _baseAssertEqual
raise self.failureException(msg)

AssertionError: <networkx.EdgeList object at 0x7f115d274200> != [(0, 1), (1, 2), (3, 4), (4, 5), (0, 3), [58 chars], 7)]

=====
Totals
=====

Ran: 1401 tests in 7.2941 sec.

- Passed: 1392
- Skipped: 7
- Expected Fail: 0
- Unexpected Success: 0
- Failed: 2

Sum of execute time for each test: 27.0069 sec.

=====
Worker Balance
=====

- Worker 0 (354 tests) => 0:00:06.893887
- Worker 1 (331 tests) => 0:00:06.631177
- Worker 2 (358 tests) => 0:00:06.497769
- Worker 3 (358 tests) => 0:00:07.294054

ERROR: InvocationError for command /home/saasha/SublimeText/rust/networkx/.tox/py/bin/steestr-run (exited with code 1)

summary

ERROR: py: commands failed

(networkx-env) saasha@Inspiron:~/SublimeText/rust/networkx/src\$