CME 212 Feedback hw1

Grading TA: yusheng@stanford.edu, GitHub ID: YosenChen

February 15, 2017

File: shortest_path.cpp
Line: 38

use min_element & lambda function

```
1  NodeIter nearest_node(const GraphType& g, const Point& point)
2  {
3    // HW1 #3: YOUR CODE HERE
4    double small_distance = norm((*g.node_begin()).position() - point);
5    auto res = g.node_begin();
6    for (auto iter = g.node_begin(); iter != g.node_end(); ++iter) {
7      double temp = norm((*iter).position() - point);
8      if (temp < small_distance) {
9        res = iter;
10       small_distance = temp;
11     }
12   }
13   return res;
14 }
```

File: shortest_path.cpp
Line: 78

default vaule is -1 for unreachable nodes

```
1  // initialize all nodes value to 0
2    for (auto ni = g.node_begin(); ni != g.node_end(); ++ni) {
3      (*ni).value() = 0;
4    }
```

File: subgraph.cpp
Line: 104

didn't create extra predicate

```
// HW1 #4: YOUR CODE HERE
// Specify and write an interesting predicate on the nodes.
// Explain what your predicate is intended to do and test it.
// If youd like you may create new nodes and tets files.
```

File: Graph.hpp
Line: 606

over-complicated

```
EdgeIterator& operator++() {
        bool jobDone = false;
        // Have not yet reached the end of the adj_list in the next step
        while (root_count_ < graph_->adj_list.size()) {
            if (!jobDone) {
                if (graph_->adj_list[root_count_].size() == 0 ) {
                    root_count_++;
                    end_count_ = 0;
                } else if(end_count_ < graph_->adj_list[root_count_].size() - 1) {
                    end_count_++;
                } else {
                    root_count_++;
                    end_count_ = 0;
                }
                jobDone = true;
            } else {
                if (graph_->adj_list[root_count_].size() == 0) {
                    root_count_++;
                    end_count_ = 0;
                    continue;
                }
                if (root_count_ <= graph_->adj_list[root_count_][end_count_]) {
                    break;
                }
                if (end_count_ < graph_->adj_list[root_count_].size() - 1) {
                    end_count_++;
                } else {
                    root_count_++;
                    end_count_ = 0;
                }
            }
        }
        position_++;
```

```
34          return *this;
35      }
```

Your hw1 grade:

8

Grade	Explanation
0	Did not try, did not hand in, or submitted too late with no late-days left.
1-2	Poor. Little to no serious attempt on this homework. Submission has barely changed since last homework (if any) or did not follow the guidelines at all.
3-4	Poor. Did not finish but a good attempt. Conveyed the message of understanding the material.
5-6	Fair. Code is buggy but could be debugged and/or some major conceptual errors. Code does work and produces output along homework guidelines.
7-8	Good. Code compiles and runs properly with mostly the right output. Some mistakes and minor conceptual errors that could be worked on.
9-10	Excellent. No or very few minor mistakes. Conveyed solid understanding of the material.
11	Exceptional. Showed extra insight. Implemented features that improved the code beyond what was requested.