UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

# Build an OpenStreetMap Route Planner

REVIEW

CODE REVIEW 11

HISTORY

## Meets Specifications

Dear student,

🎉 **Congratulations** 🎉

on successfully completing this project. The way the entire algorithm was implemented as well as the precautions that were taken to ensure that this submission runs with success is quite commendable. That was some serious hard work. If you still think there is something you didn't get, please go through the resources once again and work it out by resetting your work and do it again and take the help of the mentors and Udacity Knowledge or books, etc. in case of stuck up 👏👏
**I have provided some suggestions on project files that you can have a look at. I hope you will find them useful! Please refer to the Code Review section**

**Things I liked the most**

- Implement the algorithm correctly
- Use lambda expression correctly
- Good knowledge of std library (know how to use reverse and sort)

**Suggestions**

- Could have a boundary check for the values entered from the user
- Could have a function to get the user input to support modularity
- Add more comments for clarity
- Use only needed local vars as not to harm stack, you can check here and here for more info
- Done the checks (comparison) for the values instead of pointers as it is much safer

in the meantime when you get free enough then please visit the site http://www.cplusplus.com/. Try exploring sites like this and you will learn a lot in the process.
Look for vector and their various member functions here
Look for unordered_map and their various member functions here

For using this, it is not mandatory to explicitly use it, you may ignore it, but it might help in case of clarifying the var scope.

Now you should post this project on **GitHub** with a very nice readme

1. Make Read Me
2. Here is my friend's blog on how to write a readme

## Compiling and Testing

✓

The project code must compile without errors using `cmake` and `make`.

**Well done!** The code compiles correctly using **CMake ..** and **make** 👏

```
[ 80%] Linking CXX executable OSM_A_star_search
[ 80%] Built target OSM_A_star_search
[ 85%] Building CXX object thirdparty/googletest/googlemock/CMakeFiles/gmock.dir/src/
gmock-all.cc.o
[ 90%] Linking CXX static library ../../../lib/libgmock.a
[ 90%] Built target gmock
[ 95%] Building CXX object thirdparty/googletest/googlemock/CMakeFiles/gmock_main.dir
/src/gmock_main.cc.o
[100%] Linking CXX static library ../../../lib/libgmock_main.a
[100%] Built target gmock_main
root@b01a4e5db360:/home/workspace/CppND-Route-Planning-Project
/build#
```

**Here are some useful links for the Make file topic**, you will need it entire life being a software engineer!

1. How to Build a CMake-Based Project
2. Introduction to CMake by Example
3. Cmake tutorial

✓

Code must pass tests that are built with the `./test` executable from the `build` directory of the project. See the project submission instructions for more details on how to run the tests.

I compiled the code with the tests using **./test** from the build directory.
I ran the test executable in the build folder, and all tests passed!

Great work 👏🏼👏🏼

```
OPEN FILE
+ BASH ×
Running main() from /home/workspace/CppND-Route-Planning-Project/thirdparty/googlete
st/googletest/src/gtest_main.cc
[==========] Running 4 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 4 tests from RoutePlannerTest
[ RUN      ] RoutePlannerTest.TestCalculateHValue
[       OK ] RoutePlannerTest.TestCalculateHValue (123 ms)
[ RUN      ] RoutePlannerTest.TestAddNeighbors
[       OK ] RoutePlannerTest.TestAddNeighbors (123 ms)
[ RUN      ] RoutePlannerTest.TestConstructFinalPath
[       OK ] RoutePlannerTest.TestConstructFinalPath (120 ms)
[ RUN      ] RoutePlannerTest.TestAStarSearch
[       OK ] RoutePlannerTest.TestAStarSearch (128 ms)
[----------] 4 tests from RoutePlannerTest (494 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test case ran. (494 ms total)
[  PASSED  ] 4 tests.
root@798862677fbe:/home/workspace/CppND-Route-Planning-Project/build#
```

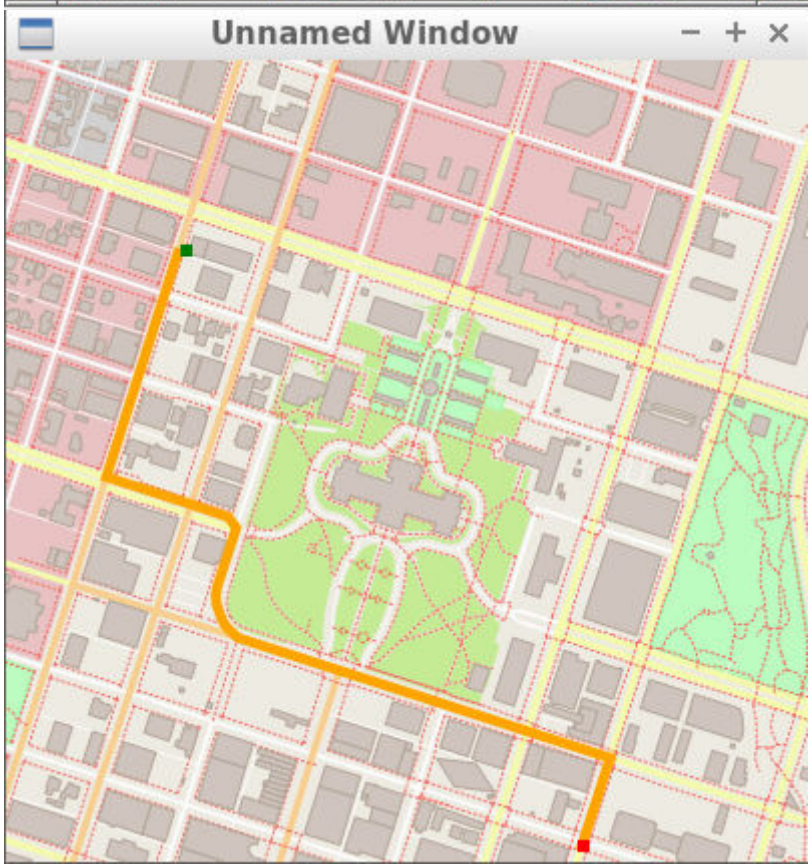
## User Input

✓

A user running the project should be able to input values between 0 and 100 for the start x, start y, end x, and end y coordinates of the search, and the project should find a path between the points.

**The distance and the routes** are correctly addressed 👏🏼

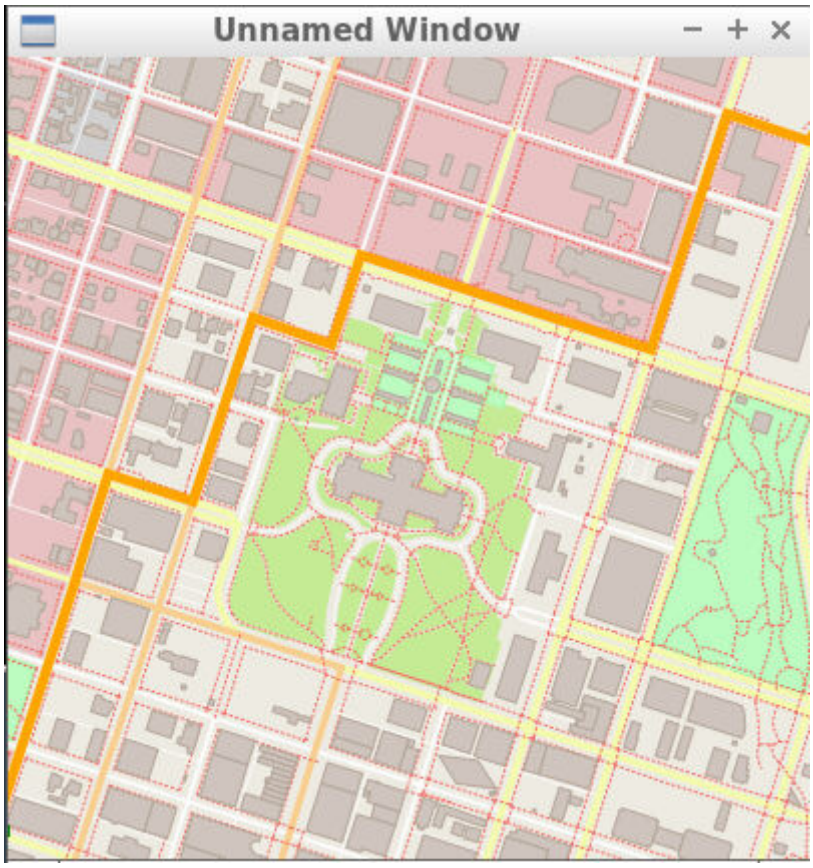

Unnamed Window


Unnamed Window

The coordinate (0, 0) should roughly correspond with the lower left corner of the map, and (100, 100) with the upper right.

Note that for some inputs, the nodes might be slightly off the edges of the map, and this is fine.

**(0, 0)** corresponds with the **lower-left** corner of the map, and **(100, 100)** with the **upper right**




## Code Efficiency

Your code does not need to sacrifice comprehension, stability, or robustness for speed. However, you should maintain good and efficient coding practices when writing your functions.

Here are some things to avoid. This is not a complete list, but there are a few examples of inefficiencies.

- Running the exact same calculation repeatedly when you can run it once, store the value and then reuse the value later.
- Loops that run too many times.
- Creating unnecessarily complex data structures when simpler structures work equivalently.
- Unnecessary control flow checks.

**Nice Job!**

- Your code is well structured and formatted!
- No more unnecessarily complex data
- No Loops that run too many times

There is a nice article on the pros and cons of commenting, click here

DOWNLOAD PROJECT

11 CODE REVIEW COMMENTS

RETURN TO PATH