



[< Return to Classroom](#)

Implement Route Planner

REVIEW

CODE REVIEW

HISTORY


Meets Specifications

Hello! I'm Sho Nakamura. I'm Japanese. I have completed this project and would appreciate a review.

Excellent work Sho Nakamura!

Congratulations on passing the project by meeting all required specifications. 🎉🎉

Good choice in using "straight line" Euclidean distance as heuristic in A-star search. Your selection of data structures is appropriate, nice work!

Keep up the good work in future projects as well! 

Reference links for learning purpose:

- [Shortest paths - A star](#)
- It is encouraged to use a heap in order to quickly find the minimum f-score at every iteration:
<https://docs.python.org/2/library/heapq.html>

Correctness

Running test.py shows "all tests pass".

All the tests have passed! Congratulations!

```
In [29]: from test import test
         test(PathPlanner)

All tests pass! Congratulations!
```

The student implements all required methods.

You have implemented all methods correctly and your submission is always finding the correct path.

Great work implementing all the methods!

The heuristic function used to estimate the distance between two intersections is guaranteed to return a distance which is less than or equal to the true path length between the intersections.

You have used the Euclidean distance for the heuristic here, which guarantees to return the minimum distance between two intersections, great!

Student answered all question correctly.

- How does A-Star search algorithm differ from Uniform cost search? ✓
- What about Best First search? ✓
- Some admissible heuristics are consistent. ✓
- All consistent heuristics are admissible. ✓

Choice and Usage of Data Structures

Code avoids obvious inappropriate use of lists and takes advantage of the performance improvement afforded by sets / dictionaries where appropriate. For example, a data structure like the "open_set" on which membership checks are frequently performed (e.g. `if node in open_set`) should not be a list.

Correct use of sets and dicts for quick membership checks and look-ups, respectively.

This item is a judgement call. Student code doesn't need to be perfect but it should avoid big performance degrading issues like...

...unnecessary duplication of lists

...looping through a large set or dictionary when a single constant-time lookup is possible

✓ Very good overall. Here are some suggestions to use list comprehension in Python and discussion on which method to call:

create_openSet: Another way of doing it:

```
return {self.start}
```

create_gScore(): You can also optimize by embedding the if .. else the value part (list comprehensions):

```
def create_gScore(self):  
    return {node: (0 if node== self.start else math.inf) for node in self.  
    map.intersections}
```

✓ create_fScore(self): Choosing to call heuristic_cost_estimate is more generic and better than calling: distance() directly

```
def create_fScore(self)  
    fScore[self.start]=self.heuristic_cost_estimate
```

 [DOWNLOAD PROJECT](#)

[RETURN TO PATH](#)