Name : Woojin Park
Email : woojinpa@andrew.cmu.edu

Search and Planning

- Written Report -

- How do the results compare to the tour returned by the Google OR code?

IDA* and Google OR shows exactly reverse order of shortest possible routes of cities. Therefore, both algorithms worked well to detect shortest path. However, if we consider that there are more than 10 cities, I can assume that the result will be vary depend on which heuristic function is more robust without overestimation. And based on the code efficiency Google OR code seems slightly better, because the length of whole code lines is way shorter and also computation speed performance was slightly better.

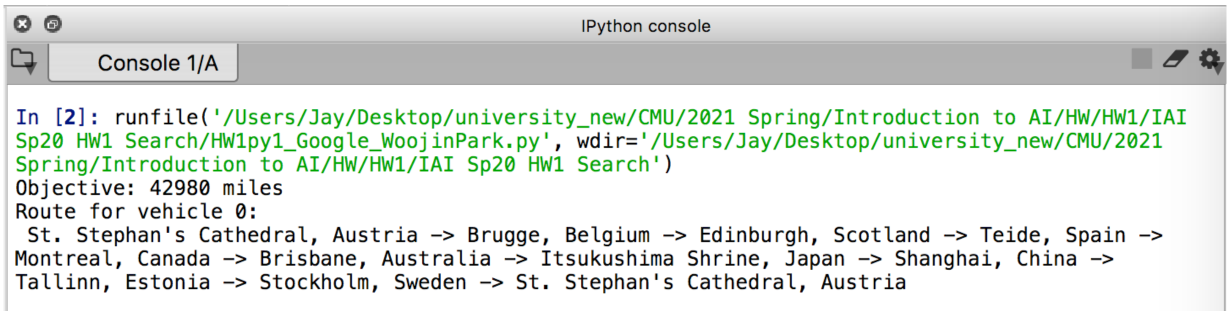
- Is IDA* a good choice for TSP? What other search algorithms might be better than IDA*?

"yes, It is a good choice for TSP." Because A* always find a cheapest solution path specifically doing best-first search and finding out the least-cost path from a given initial node to the goal node (out of one or more possible goals). But only if IDA* heuristic function is admissible and consistent. But if we think about other algorithms might perform better than A*, the "Naive approach" can be the one possible option. Because it guarantees to find the exact solution in a short amount of time,
the Nearest Neighbor (NN) approximation algorithm attempts to find a decent solution in as little time as possible. When we start searching at a random city, this algorithm compares all possible permutations of paths to discover the shortest unique solution. And also the processing is relatively shorter than any other algorithms. Therefore, we always need to cross validate the performance of each algorithm by given number of cities.

Name : Woojin Park
Email : woojinpa@andrew.cmu.edu

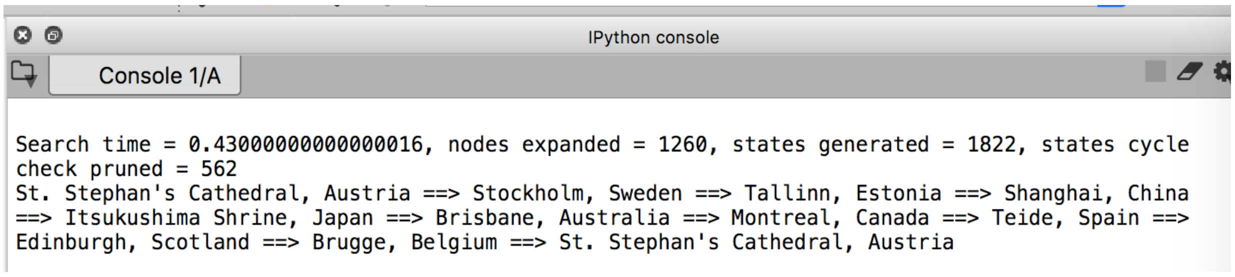
- Result Screenshot -

Part 1: Using ORTools to implement TSP



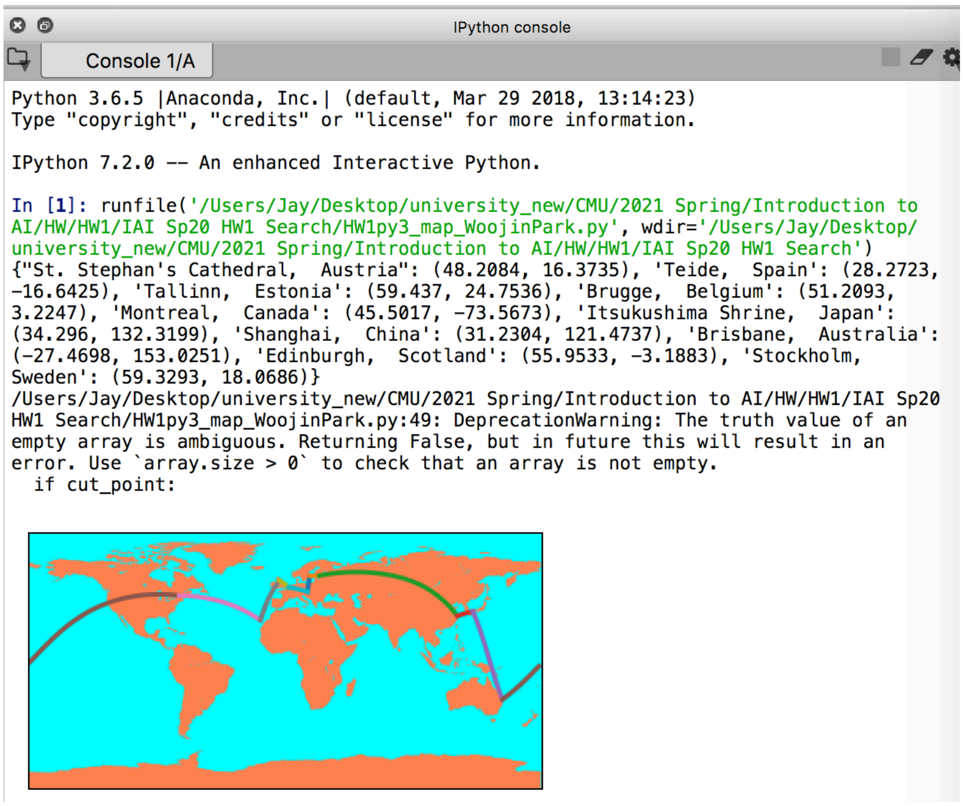
```
In [2]: runfile('/Users/Jay/Desktop/university_new/CMU/2021 Spring/Introduction to AI/HW/HW1/IAI
Sp20 HW1 Search/HW1py1_Google_WoojinPark.py', wdir='/Users/Jay/Desktop/university_new/CMU/2021
Spring/Introduction to AI/HW/HW1/IAI Sp20 HW1 Search')
Objective: 42980 miles
Route for vehicle 0:
 St. Stephan's Cathedral, Austria -> Brugge, Belgium -> Edinburgh, Scotland -> Teide, Spain ->
Montreal, Canada -> Brisbane, Australia -> Itsukushima Shrine, Japan -> Shanghai, China ->
Tallinn, Estonia -> Stockholm, Sweden -> St. Stephan's Cathedral, Austria
```

Part 2: Solve the TSP using your own implementation of IDA* Algorithm



```
Search time = 0.43000000000000016, nodes expanded = 1260, states generated = 1822, states cycle
check pruned = 562
St. Stephan's Cathedral, Austria ==> Stockholm, Sweden ==> Tallinn, Estonia ==> Shanghai, China
==> Itsukushima Shrine, Japan ==> Brisbane, Australia ==> Montreal, Canada ==> Teide, Spain ==>
Edinburgh, Scotland ==> Brugge, Belgium ==> St. Stephan's Cathedral, Austria
```

Part 3: Plotting the tour on world map



```
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:14:23)
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/Jay/Desktop/university_new/CMU/2021 Spring/Introduction to
AI/HW/HW1/IAI Sp20 HW1 Search/HW1py3_map_WoojinPark.py', wdir='/Users/Jay/Desktop/
university_new/CMU/2021 Spring/Introduction to AI/HW/HW1/IAI Sp20 HW1 Search')
{"St. Stephan's Cathedral,  Austria": (48.2084, 16.3735), 'Teide,  Spain': (28.2723,
-16.6425), 'Tallinn,  Estonia': (59.437, 24.7536), 'Brugge,  Belgium': (51.2093,
3.2247), 'Montreal,  Canada': (45.5017, -73.5673), 'Itsukushima Shrine,  Japan':
(34.296, 132.3199), 'Shanghai,  China': (31.2304, 121.4737), 'Brisbane,  Australia':
(-27.4698, 153.0251), 'Edinburgh,  Scotland': (55.9533, -3.1883), 'Stockholm,
Sweden': (59.3293, 18.0686)}
/Users/Jay/Desktop/university_new/CMU/2021 Spring/Introduction to AI/HW/HW1/IAI Sp20
HW1 Search/HW1py3_map_WoojinPark.py:49: DeprecationWarning: The truth value of an
empty array is ambiguous. Returning False, but in future this will result in an
error. Use `array.size > 0` to check that an array is not empty.
  if cut_point:
```