# Recursion and Debugger-like Features for METAL Algorithm Visualizations

Alissa Ronca, Tyler Gorman, Zachary Goodsell, James D. Teresco (advisor)

## Introduction

**What is METAL?**

- Map-based Educational Tools for Algorithm Learning (METAL) is an ongoing project that aims to help students learn algorithms more efficiently with Algorithm Visualization (AV).
- This AV system allows a student to interact with one of several algorithms in action on highway data, following changes in key variables and data structures with corresponding color-coded changes to the data on the map.
- By running on data of various sizes and simulating at various speeds, a student can gain a deeper understanding more quickly.

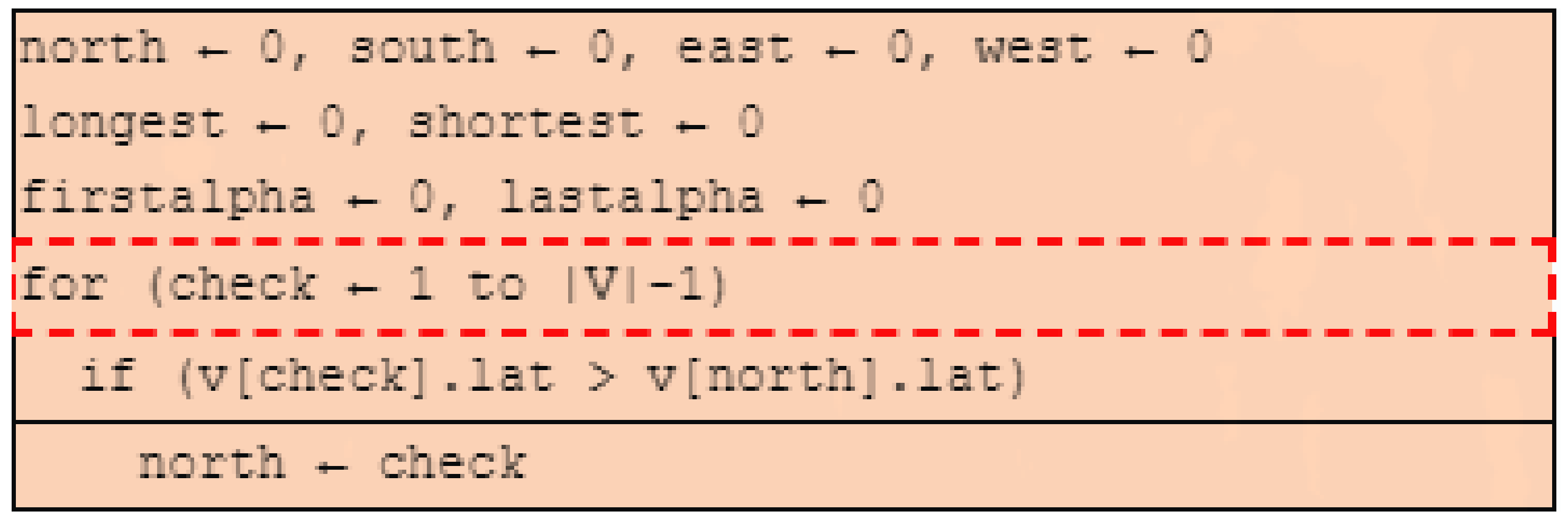
## Summer 2019

**Improvements to the project**

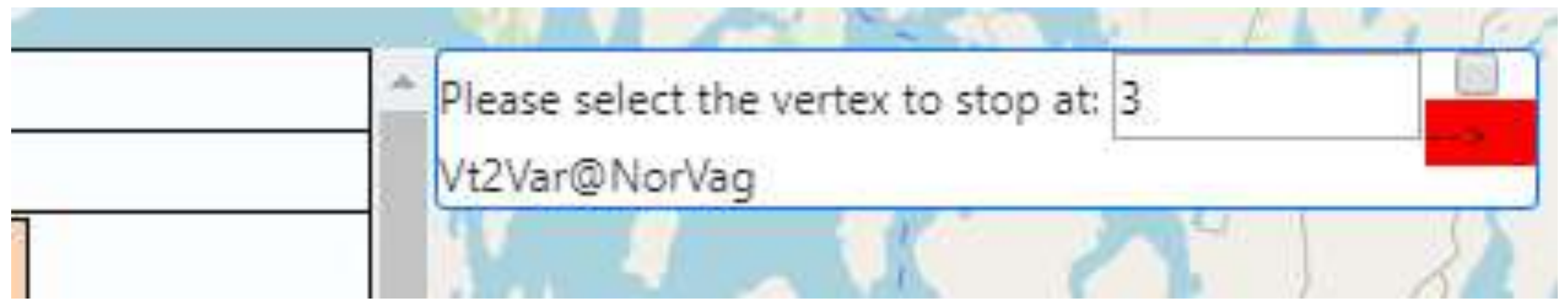
- Our Summer 2019 work on METAL focused on enhancing its ability to represent recursion and debugger-like features.

## Debugger-like Features

METAL AVs supports both unconditional and conditional breakpoints, giving debugger-like capabilities. Suppose you wanted to run an algorithm until a particular line of code is executed, or even until a specific waypoint is first visited, or a specific connection is first added to or removed from some data structure. Breakpoints provide these capabilities. To set a breakpoint, click on a line of code. If it has breakpoint capabilities (the vast majority of lines do), it will become highlighted with a red-dashed border.



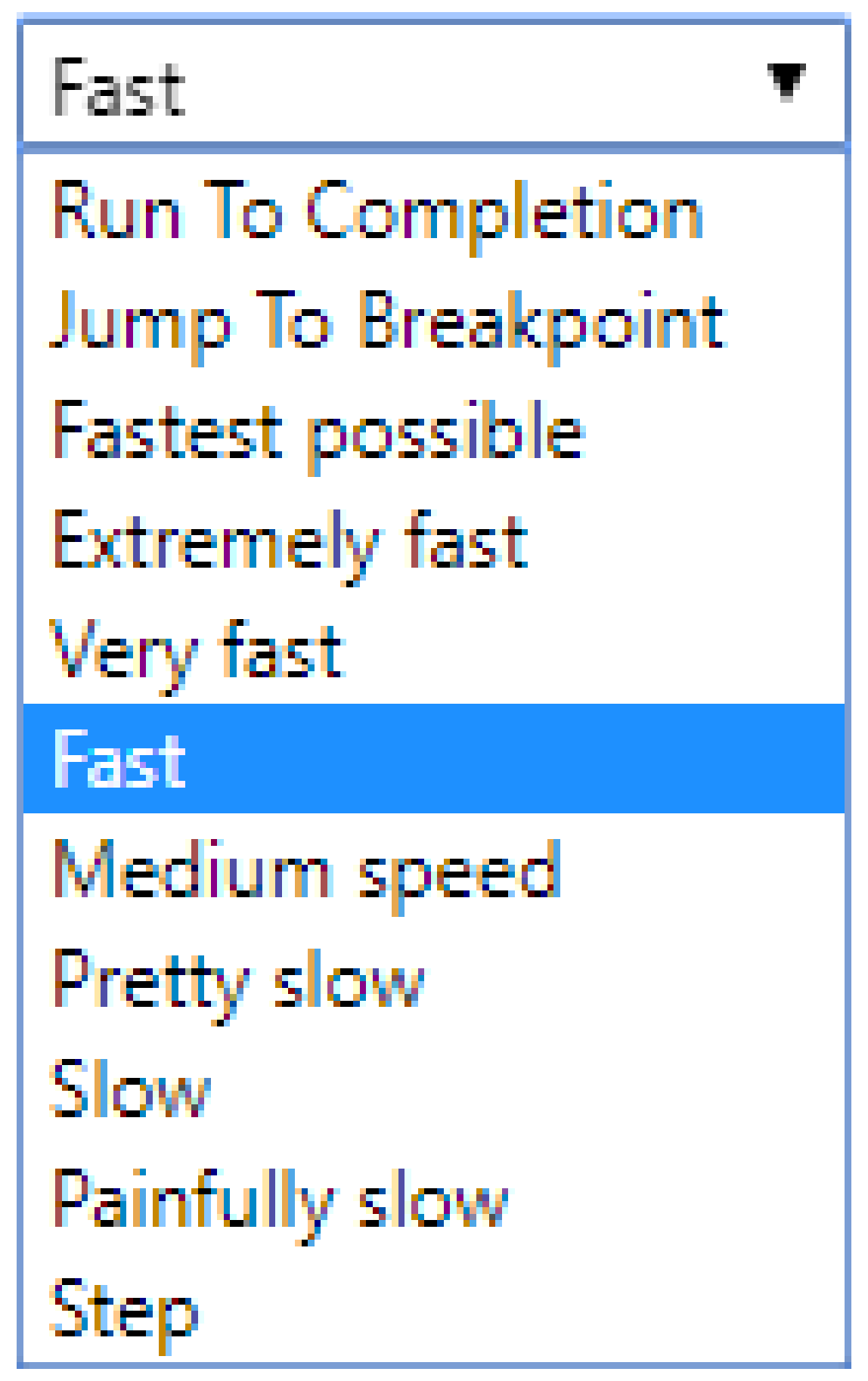
```
north ← 0, south ← 0, east ← 0, west ← 0
longest ← 0, shortest ← 0
firstalpha ← 0, lastalpha ← 0
for (check ← 1 to |V|-1)
  if (v[check].lat > v[north].lat)
    north ← check
```



Please select the vertex to stop at: 3

Vt2Var@NorVag

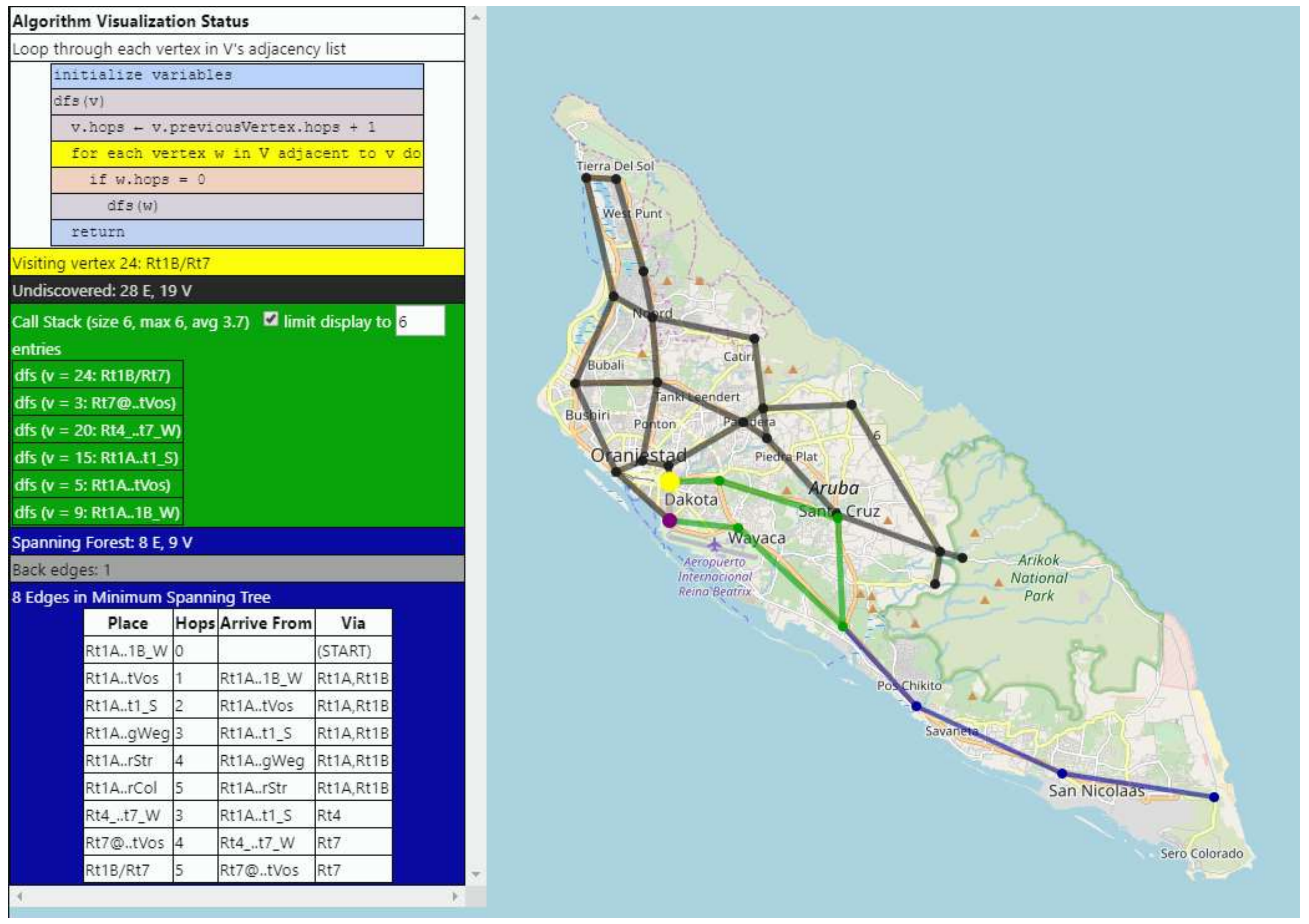
When the AV is executed, it will stop every time it encounters that line of code, so you can explore the state of the algorithm's progress in the AV Status Panel, on the map, and in the data tables. Most lines of code also support conditional breakpoints. This means that execution would only stop on those lines if certain conditions are met. When a breakpoint is set at a line that supports conditional breakpoints, a small red arrow appears to the right of the AV Status Panel.



These are algorithm-specific, but common examples include stopping a for loop at a specific vertex, stopping only when the condition of an if statement takes on a specific value, or the insertion or removal of specific values from a data structure. Unconditional and conditional breakpoints are honored with any speed setting except "Run to Completion". To execute as quickly as possible to the next breakpoint, the "Jump to Breakpoint" setting can be used.

## Recursion

- The algorithm that was successfully implemented is the depth-first search.
- Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node and explores as far as possible along each branch before backtracking.
- The grey lines are undiscovered route on the map.
- As the routes are discovered they turn green and are added to the spanning tree.
- As the algorithm backtracks the routes become blue to signify the algorithm no longer using this route.
- The yellow dot show the current vertex that the algorithm is visiting.
- The purple node shows the start point for the algorithm.



**Algorithm Visualization Status**

Loop through each vertex in V's adjacency list

```
initialize variables
dfs(v)
  v.hops ← v.previousVertex.hops + 1
  for each vertex w in V adjacent to v do
    if w.hops = 0
      dfs(w)
  return
```

Visiting vertex 24: Rt1B/Rt7

Undiscovered: 28 E, 19 V

Call Stack (size 6, max 6, avg 3.7) ✓ limit display to 6 entries

dfs (v = 24: Rt1B/Rt7)
dfs (v = 3: Rt7@..tVos)
dfs (v = 20: Rt4_..t7_W)
dfs (v = 15: Rt1A..t1_S)
dfs (v = 5: Rt1A..tVos)
dfs (v = 9: Rt1A..1B_W)

Spanning Forest: 8 E, 9 V

Back edges: 1

8 Edges in Minimum Spanning Tree

| Place | Hops | Arrive From | Via |
|---|---|---|---|
| Rt1A..1B_W | 0 | | (START) |
| Rt1A..tVos | 1 | Rt1A..1B_W | Rt1A,Rt1B |
| Rt1A..t1_S | 2 | Rt1A..tVos | Rt1A,Rt1B |
| Rt1A..gWeg | 3 | Rt1A..t1_S | Rt1A,Rt1B |
| Rt1A..rStr | 4 | Rt1A..gWeg | Rt1A,Rt1B |
| Rt1A..rCol | 5 | Rt1A..rStr | Rt1A,Rt1B |
| Rt4_..t7_W | 3 | Rt1A..t1_S | Rt4 |
| Rt7@..tVos | 4 | Rt4_..t7_W | Rt7 |
| Rt1B/Rt7 | 5 | Rt7@..tVos | Rt7 |

## Future Work

Continue developing and implementing more recursive algorithms for the METAL project.

## Acknowledgements

- The Summer Scholars 2019 project was funded by the Center for Undergraduate Research and Creative Activities (CURCA) at Siena College.
- METAL's web site includes highway data and AV capabilities in the Highway Data Examiner (http://courses.teresco.org/metal).
- Further details about METAL and its use as a pedagogical tool can be found on the web site and in the publications listed there.
- METAL's highway data is derived from the Travel Mapping Project (http://travelmapping.net).

SCAN ME

SIENA college SUMMER SCHOLARS Center for Undergraduate Research and Creative Activity