# CSE 512 LABORATORY – Week 3, Winter 2016

**Prof. Kerstin Voigt**

In this lab, we will work through several modifications of the path finding functions in the `path_finder.py` module.

Start by obtaining a fresh copy of

```
~voigt/cse512/path_finder.py
```

Load the program and try the functions for serveral pairs of nodes in the A-to-Q graph stored in the global variable `TheMap`.

Also obtain a copy of the `BotGridPlain.pdf` file (in ∼`voigt/cse512` or a lecture handout). Whereas `TheMap` represents a **directed, acyclic** graph, the graph in `BotGridPlain.pdf` is not directed. This means that there is the unfortunate potential for cycle when navigation through the graph. With a few straight-forward modications, our path finding code can be upgraded to one that will navigate undirected graphs and actively avoid running in cycles.

Work on the following exercises in sequence:

1. Change the global variable `TheMap` so that it captures the map depicted in the `BotGridPlain.pdf` handout.
2. Add code to function `all_paths(x,y)` that will print out each element as it is taken off list `open`. Also add a counting variable that keeps track of the number of iterations of the `while`-loop. Add another line of code that will terminate the loop after 500 iterations.
3. Add a "switch" parameter to `all_paths` that will allow us to run the function in two modes: (a) return first path found, and (b) return all paths, the original mode. Make a small change in the function that will have it respond to both switch settings.
4. Run the new version of `all_paths` and observe its behavior.
5. Your instructor will suggest severall other modification and ask to explore them. Likely this will lead to some discussion ....
6. Add two more `print` statements to `all_paths` which print out the contents of local variables `open` and `closed` for each iteration. What do you see?

7. Start to work on another variant of `all_paths` which will store on `open` and `closed` not the paths of nodes (list structures) but elements of type `Node`. For this, you will need to define your own `class Node`. An object of type `Node` should have the following data member:

   - a label (single character of string)
   - the parent node which is the node "from" which the node was reached

   Follow up with all code modifications and additions that are needed to have the new variant of `all_paths` perform like the original.

8. What speaks in favor of the variant of `all_paths` that works with a `class Node`?

**Again, consider the completion of this lab exercise your next homework assig- ment.** It is due on **Thursday, Feb 4, at the time of the lecture.** Hand in a hardcope of your code and a typescript (copy of the relevant section of your idle interpreter window). With your typescript, you should demonstrate that your `all_paths` function using `class Node` successfully finds **one** path from A to Q. When a path exists, your program should also print out the sequence of nodes that make up the path it found. Hint: the information can be extracted from the contents of variable `closed`.