# ICS 340 Programming Project, Deliverable C

***Specification:***

Start with your (ideally working) submission of Deliverable B. Read a file of the name “F[<whatever>]c.txt. This is a file of distances between cities. Note that it won’t be a complete graph, unlike the case for Deliverable B.

From the starting city (val = “S”) use depth first search with iterative deepening to find the nearest goal city (val = “G”). Start with a depth bound of 1, and increase the depth bound by 1 each time through the graph. Print out your results for each depth bound that causes a change over the previous depth bound.

While doing the DFS, if you have a choice of two nodes to visit, visit the nearer node first. In case of a tie, break ties alphabetically.

When your program finishes, print the path to the nearest goal node, and its distance.

The “prog340” handout describes the format of the input file for this and all program deliverables.

As will always be the case in this class, the program must be written in Java and must run on the University Windows computer systems. To ensure this I strongly recommend that you:

1. Use only Oracle Java 13 SE and earlier constructs, and
2. Test it on the University systems before submission if you have any doubts about its ability to run on the University Windows.

Submit the Java source code to the open Deliverable A submission folder. You may submit either the source code or a full Eclipse package.

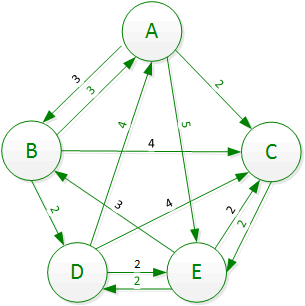
**Algorithm:**

The iterative deepening algorithm with depth first search is well described in the notes and in the Poole text, Chapter 3.

**Output:**

Here are a couple small graphs with the iterative deepening DFS as I am looking for.

First graph: Start = A, Goal = E



Yields output:

Depth 1:

A

Depth 2:

ACA

Depth 3:

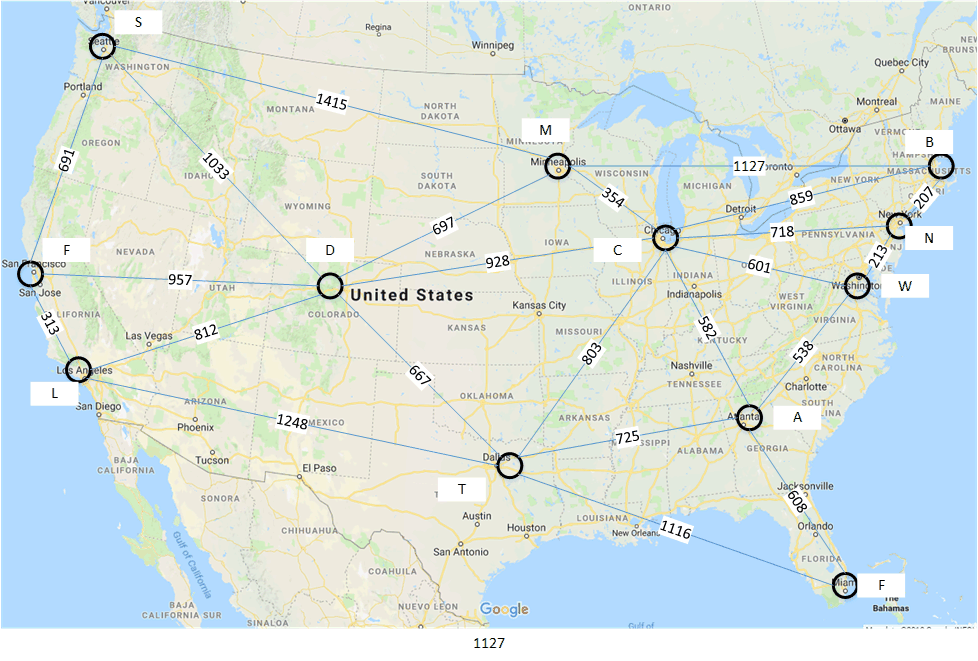
ACABA

Depth 4:

ACE

Path found: ACE, length = 4

Second graph: Start = S, Goal = D

Yields output:

Depth 1:

S

Depth 691:

SFS

Depth 1004:

SFLFS



Depth 1033:

SFLFSD

Path found: SD, length = 1033

**Submit:**

Submit your code as an Eclipse package, or submit all the “.java” source files in a zipped archive. Do not include test files.

**Test Files:**

Three test files are included on D2L (F1c, F2c, F3c). More may be added. I may also test with other files that you haven’t seen.

**Grading:**

This deliverable is worth 75 points: Correctness will be assessed for 4-6 cases. The number of points will be higher for more complex cases. There will be 5 additional points for design and 5 for documentation. There will be one 5-point regression test of deliverable A and one 5-point regression test of deliverable B.

**Due Dates:**

The program is due on Saturday, February 20th at noon for full credit in the D2L “Deliverable B” dropbox. For 80% of credit earned, you may (re)submit it by noon Saturday, March 6th (late submission possible with late penalties). The time of submission is the time that D2L lists the file as submitted.

**Comments:**

* I think that getting the distance is not much harder than doing deliverable A, but keeping track of the cities in order to print the optimal tour is tougher. Basically, if you can get the correct distances that’s a B+ even if you don’t get the cities listed in the right order.
* Try to use the same or similar notation to one of the papers in your program, it makes it easier for you to figure out what you’re doing and for me to grade it.