MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.01—Introduction to EECS I
Spring Semester, 2008

# Software lab 10, Issued Tuesday, April 15

## Overview of this week's work

### In software lab

- Work through the software lab.

### Before the start of your design lab on Apr 17 or 18

- Read the class notes and review the lecture handout.
- Do the on-line tutor problems in section 10.1.

### In design lab

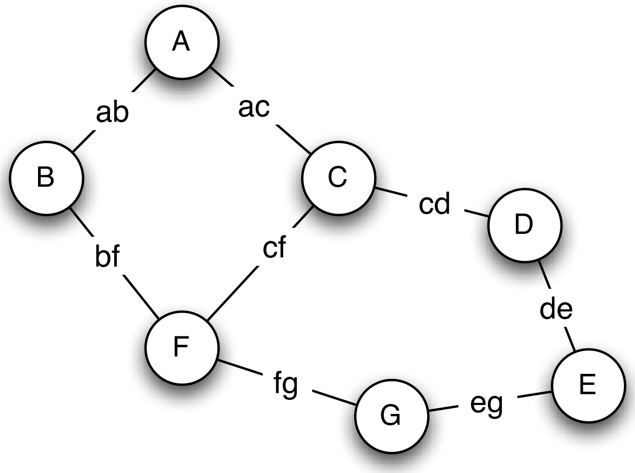
- Do the nano-quiz.
- Work through the design lab.

### At the beginning of your next software lab on Apr 15 or 16

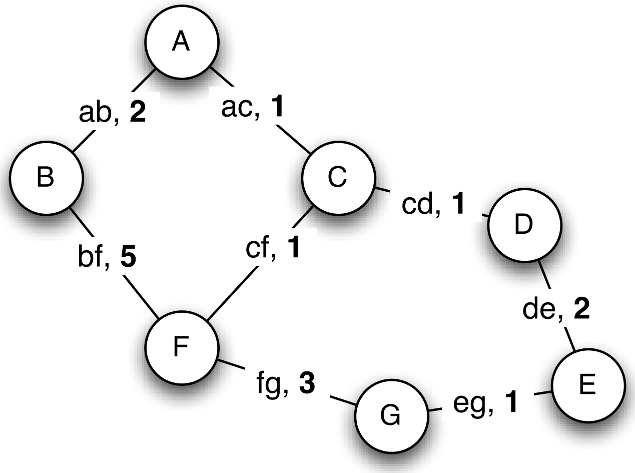
- Submit written solutions to questions 3—6 from the software lab. See the design lab handout for a specification of what to turn in from there. All written work must conform to the homework guidelines on the web page.

## Software Lab: Search

The code file `search.py` contains the code for the search algorithms, city graphs, and the `numberTest` domain discussed in lecture. Load this into IDLE so you can experiment with it.



A.



B.

Figure 1: A. A simple map. B. A map with each road labeled by the time it takes to traverse it.

You can set the global variable `somewhatVerbose` to `True` to see some of what is going on in the search process. You can set `verbose` to `True` to see every step in detail (more than you will usually want).

**Question** 1: Consider the graph in figure 1A. Imagine it's a road network between cities with short names.

a. Formulate a successor function for this domain. It should take a state as input, and return a list of action-state pairs.
b. Formulate a goal test function.
c. Use your successor and goal-test functions together with the `search` function to find the path from A to E using each of our four search methods (breadth and depth search, with and without dynamic programming). (Your results will vary depending on the order in which you create the successors, if this example does not produce differences among the different methods, pick a start and goal that do).

**Question** 2: Now, look at the graph in figure 1B. It's the same road network, but now the arcs are labeled with an amount of time it takes to traverse them, as well as the action names. We'd like to make plans to traverse this network, but with the goal of arriving at a particular node at a particular time, for example, reaching node F exactly at time 8, assuming we started at node A at time 0. What is an appropriate choice of state space for this planning problem?

**Question** 3: Implement a successor and goal-test functions for this problem. Run the four different kinds of searches on this problem. Report the results. (Include code and results in your write-up).

**Question** 4: What happens to the search when you ask for a goal that is impossible, for example, leaving node A at time 0 and arriving at node G at time 4? Is it possible to arrive at F (from A) exactly at time 3? How about exactly at time 4? How does the code deal with these cases?

**Question** 5: Draw out the complete search tree (as in the course notes) starting at A for the graph in figure 1A (the first version of the problem, without time). Use "Pruning Rule 1", that is, don't consider any path that visits a city more than once. How many nodes are there in the whole tree? Indicate which nodes are expanded by depth-first search and which by breadth first search (both without dynamic programming). Indicate which of these nodes are **not** expanded when using dynamic programming. Explain the results (final path, nodes expanded, nodes visited) you saw in the previous question in terms of this tree.

**Question** 6: How big can the search trees possibly get for the version of the problem with time added? Does dynamic programming help much for this problem? Explain.