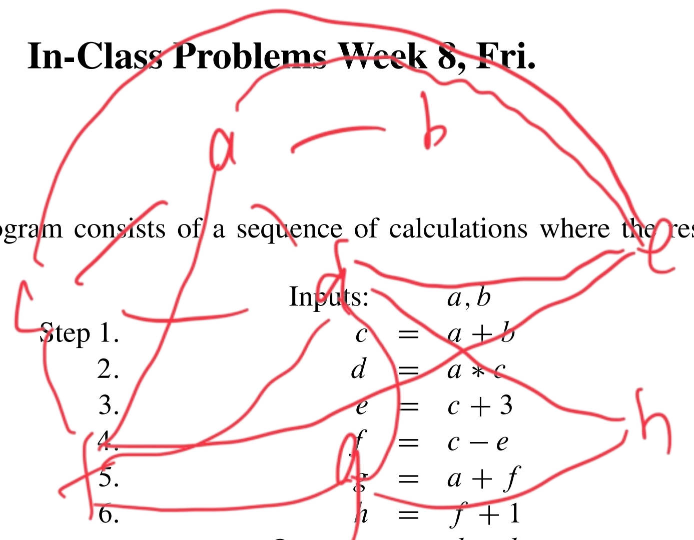
In-Class Problems Week 8, Fri.



Problem 1.
A portion of a computer program consists of a sequence of calculations where the results are stored in variables, like this:

$$\begin{array}{rrcl} & \text{Inputs:} & & a, b \\ \text{Step 1.} & c & = & a+b \\ 2. & d & = & a*c \\ 3. & e & = & c+3 \\ 4. & f & = & c-e \\ 5. & g & = & a+f \\ 6. & h & = & f+1 \\ & \text{Outputs:} & & d, g, h \end{array}$$

A computer can perform such calculations most quickly if the value of each variable is stored in a *register*, a chunk of very fast memory inside the microprocessor. Programming language compilers face the problem of assigning each variable in a program to a register. Computers usually have few registers, however, so they must be used wisely and reused often. This is called the *register allocation* problem.

In the example above, variables a and b must be assigned different registers, because they hold distinct input values. Furthermore, c and d must be assigned different registers; if they used the same one, then the value of c would be overwritten in the second step and we d get the wrong answer in the third step. On the other hand, variables b and d may use the same register; after the first step, we no longer need b and can overwrite the register that holds its value. Also, f and h may use the same register; once $f+1$ is evaluated in the last step, the register holding the value of f can be overwritten.

(a) Recast the register allocation problem as a question about graph coloring. What do the vertices correspond to? variables Under what conditions should there be an edge between two vertices? Construct the graph corresponding to the example above. can be in same registers

(b) Color your graph using as few colors as you can. Call the computer s registers $R1$, $R2$, etc. Describe the assignment of variables to registers implied by your coloring. How many registers do you need?

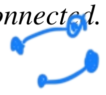
(c) Suppose that a variable is assigned a value more than once, as in the code snippet below:

$$\begin{array}{l} \ldots \\ t_1 \quad t = r + s \\ u = t * 3 \\ t_2 \quad t = m - k \\ v = t + u \\ \ldots t_2 \end{array}$$

How might you cope with this complication?

Problem 2.

False Claim. *If every vertex in a graph has positive degree, then the graph is connected.*



(a) Prove that this Claim is indeed false by providing a counterexample.

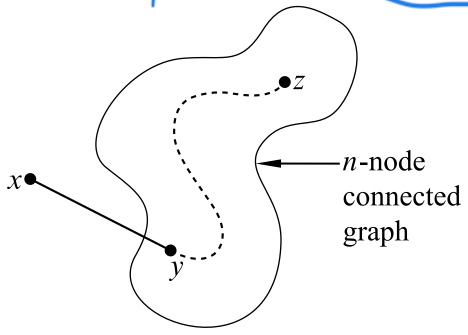
(b) Since the Claim is false, there must be a logical mistake in the following bogus proof. Pinpoint the *rst* logical mistake (unjustified step) in the proof.

Bogus proof. We prove the Claim above by induction. Let $P(n)$ be the proposition that if every vertex in an n-vertex graph has positive degree, then the graph is connected.

Base cases: ($n \leq 2$). In a graph with 1 vertex, that vertex cannot have positive degree, so $P(1)$ holds vacuously.

$P(2)$ holds because there is only one graph with two vertices of positive degree, namely, the graph with an edge between the vertices, and this graph is connected.

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 2$. Consider an n-vertex graph in which every vertex has positive degree. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now we add one more vertex x to obtain an $(n+1)$-vertex graph:



All that remains is to check that there is a path from x to every other vertex z. Since x has positive degree, there is an edge from x to some other vertex, y. Thus, we can obtain a path from x to z by going from x to y and then following the path from y to z. This proves $P(n+1)$.

By the principle of induction, $P(n)$ is true for all $n \geq 0$, which proves the Claim.

■

Problem 3.
In this problem, we examine an interesting connection between propositional logic and 3-colorings of certain special graphs. Consider the graph in Figure 1. We designate the vertices connected in the triangle on the left as *color-vertices*; since they form a triangle, they are forced to have different colors in any coloring of the graph. The colors assigned to the color-vertices will be called **T**, **F** and **N**. The dotted lines indicate edges to the color-vertex **N**.

(a) Prove that there exists a 3-coloring of the graph iff neither P nor Q are colored N.

(b) Argue that the graph in Figure 1 acts like a two-input OR-gate: a valid 3-coloring of the graph has the vertex labelled (P OR Q) colored **T** iff at least one of the vertices labelled P and Q are colored **T**.

(c) Changing the endpoint of one edge in Figure 1 will turn it into a two-input AND simulator. Explain.

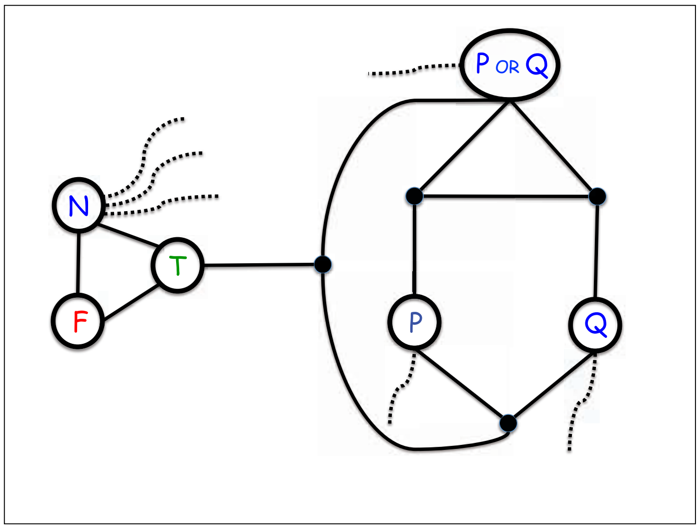


Figure 1 A 3-color OR-gate

Problem 4.
The n-dimensional *hypercube*, H_n, is a graph whose vertices are the binary strings of length n. Two vertices are adjacent if and only if they differ in exactly 1 bit. For example, in H_3, vertices 111 and 011 are adjacent because they differ only in the first bit, while vertices 101 and 011 are not adjacent because they differ at both the first and second bits.

$\overline{x_1x_2x_3}$ $\overline{y_1y_2y_3}$

(a) Verify that for any two vertices $x \neq y$ of H_3, there are 3 paths from x to y in H_3, such that, besides x and y, no two of those paths have a vertex in common.

(b) Conclude that the connectivity of H_3 is 3.

(c) Try extending your reasoning to H_4. (In fact, the connectivity of H_n is n for all $n \geq 1$. A proof appears in the problem solution.)

(a) $x_i \neq y_i$ $i=1,2,3$

$x_1 = y_1$ $\overline{x_1x_2x_3} \to x_1y_2x_3 \to x_1y_2y_3$

$x_1x_2x_3 \to x_1x_2y_3 \to x_1y_2y_3$

$x_1 = y_1$
$x_2 = y_2$

$x_1x_2x_3 \to x_1'x_2x_3$ $x_1'y_2y_3 \to x_1y_2y_3$

$\downarrow$ $\nearrow$

$x_1'y_2x_3$