Staff Solutions to In-Class Problems Week 8, Mon.

STAFF NOTE: Simple Graphs: Bipartite Matching & Coloring, Ch. 12.5-12.6

Problem 1.
A certain Institute of Technology has a lot of student clubs; these are loosely overseen by the Student Association. Each eligible club would like to delegate one of its members to appeal to the Dean for funding, but the Dean will not allow a student to be the delegate of more than one club. Fortunately, the Association VP took Math for Computer Science and recognizes a matching problem when she sees one.

(a) Explain how to model the delegate selection problem as a bipartite matching problem.

Solution. Define a bipartite graph with the student clubs as one set of vertices and everybody who belongs to some club as the other set of vertices. Let a club and a student be adjacent exactly when the student belongs to the club. Now a matching of clubs to students will give a proper selection of delegates: every club will have a delegate, and every delegate will represent exactly one club.

(b) The VP’s records show that no student is a member of more than 9 clubs. The VP also knows that to be eligible for support from the Dean’s office, a club must have at least 13 members. That’s enough for her to guarantee there is a proper delegate selection. Explain. (If only the VP had taken an Algorithms class, she could even have found a delegate selection without much effort.)

Solution. The degree of every club is at least 13, and the degree of every student is at most 9, so the graph is degree-constrained, which implies there will be no bottlenecks to prevent a matching. Hall’s Theorem then guarantees a matching.

Problem 2.
A portion of a computer program consists of a sequence of calculations where the results are stored in variables, like this:

\[
\begin{align*}
\text{Inputs:} & \quad a, b \\
\text{Step 1.} & \quad c = a + b \\
\text{2.} & \quad d = a \times c \\
\text{3.} & \quad e = c + 3 \\
\text{4.} & \quad f = c - e \\
\text{5.} & \quad g = a + f \\
\text{6.} & \quad h = f + 1 \\
\text{Outputs:} & \quad d, g, h
\end{align*}
\]

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.

2015, Albert R Meyer. This work is available under the terms of the Creative Commons Attribution-ShareAlike 3.0 license.
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? Under what conditions should there be an edge between two vertices? Construct the graph corresponding to the example above.

Solution. There is one vertex for each variable. An edge between two vertices indicates that the values of the variables must be stored in different registers. We can tell when two variables must be stored in different registers as follows: classify each appearance of a variable in the program as either an assignment or a use. An appearance is an assignment when the variable is on the left side of an equation or on the “Inputs” line. An appearance of a variable is a use if the variable is on the right side of an equation or on the “Outputs” line. The lifetime of a variable is the segment of code extending from the initial assignment of the variable until the last use. There is an edge between two variables iff their lifetimes overlap.\(^1\) This rule generates the following graph:

(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?

Solution. Four registers are needed.

One possible assignment of variables to registers is indicated in the figure above. In general, coloring a graph using the minimum number of colors is quite difficult; no efficient procedure is known. However, the register allocation problem always leads to an interval graph, and optimal colorings for interval graphs are always easy to find. This makes it easy for compilers to allocate a minimum number of registers.\(^1\)

---

\(^1\)This specification of edges is for the case that each variable is assigned at most once (see part (c)).

We are also assuming that all variables are relevant to the Outputs, where a variable is relevant iff it is an Output or is used in an assignment to a relevant variable. This is a recursive—not a circular—definition of relevant variable!

Likewise, we assume that all variables are dependent on the Inputs, where a variable is dependent on the Inputs iff it is an Input or appears in the left hand side of an assignment whose right hand side contains a dependent variable.
(c) Suppose that a variable is assigned a value more than once, as in the code snippet below:

\[
\begin{align*}
\ldots \\
& t = r + s \\
& u = t \times 3 \\
& t = m - k \\
& v = t + u \\
\ldots 
\end{align*}
\]

How might you cope with this complication?

**Solution.** Each time a variable is reassigned, we could regard it as a completely new variable. Then we would regard the example as equivalent to the following:

\[
\begin{align*}
\ldots \\
& t = r + s \\
& u = t \times 3 \\
& t' = m - k \\
& v = t' + u \\
\ldots 
\end{align*}
\]

We can now proceed with graph construction and coloring as before.

**Problem 3.**

In this problem, we examine an interesting connection between propositional logic and 3-colorings of certain special graphs. In the graph shown in Figure 1, designate the vertices connected in the triangle on the left as color-vertices. Since each color-vertex is adjacent to the other two, they must 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.

**Solution.** (left-to-right case): If there is a valid 3-coloring (or more generally, any valid coloring) then the dotted edges ensure that P and Q are not colored as N in that coloring.

(right-to-left case): If neither P nor Q are colored N, then both P and Q have to be colored T or F.

The diagram is symmetric in P and Q, so there are really only three cases to consider: P and Q are both colored T, both colored F, or P and Q are colored differently. If P and Q are colored differently, we can verify that this leads to only one possible 3-coloring where the vertex labelled \((P \lor Q)\) is colored T.

If P and Q have the same color, then one of the vertices directly above must be colored with N and the other with the opposite color as P and Q. This forces \((P \lor Q)\) to be colored with the same color as P and Q. There is then a unique coloring of the bottom vertex, and the middle vertex on the arc on the left that can complete a 3-coloring.

Therefore, in each case where neither P nor Q are colored N, there exists a valid 3-coloring.

(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 \((P \lor Q)\) colored T iff at least one of the vertices labelled P and Q are colored T.
Figure 1  A 3-color OR-gate

Solution. We can think of $P$ and $Q$ as “input” vertices and $(P \lor Q)$ as the “output” vertex. In the argument above, we concluded that when $P$ and $Q$ have different colors, $(P \lor Q)$ is colored $T$. On the other hand, when $P$ and $Q$ have the same color, then $(P \lor Q)$ also shares this color. Therefore, the color of $P \lor Q$ is always the Boolean OR of the colors assigned to $P$ and $Q$.

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

Solution. Change the endpoint of the horizontal edge incident to the $T$-vertex to be incident to the $F$ vertex. As before, when $P$ and $Q$ have the same color, the “$(P \lor Q)$”-vertex must be colored with the same color. Likewise, if $P$ and $Q$ are colored differently, then the bottommost vertex must be colored $N$ which forces the leftmost black vertex, which is now incident to the $F$ vertex, to be colored $T$, forcing the “$(P \lor Q)$”-vertex to be colored $F$.

Hence, with this edge change, the “$(P \lor Q)$”-vertex is now really a $(P \land Q)$-vertex.

Problem 4.

False Claim. Let $G$ be a graph whose vertex degrees are all $\leq k$. If $G$ has a vertex of degree strictly less than $k$, then $G$ is $k$-colorable.

(a) Give a counterexample to the False Claim when $k = 2$.

Solution. One node by itself, and a separate triangle ($K_3$). The graph has max degree 2, and a node of degree zero, but is not 2-colorable.

(b) Underline the exact sentence or part of a sentence that is the first unjustified step in the following bogus proof of the False Claim.
Bogus proof. Proof by induction on the number $n$ of vertices:

The induction hypothesis, $P(n)$ is:

Let $G$ be an $n$-vertex graph whose vertex degrees are all $\leq k$. If $G$ also has a vertex of degree strictly less than $k$, then $G$ is $k$-colorable.

**Base case:** ($n = 1$) $G$ has one vertex, the degree of which is 0. Since $G$ is 1-colorable, $P(1)$ holds.

**Inductive step:** We may assume $P(n)$. To prove $P(n + 1)$, let $G_{n+1}$ be a graph with $n + 1$ vertices whose vertex degrees are all $k$ or less. Also, suppose $G_{n+1}$ has a vertex, $v$, of degree strictly less than $k$. Now we only need to prove that $G_{n+1}$ is $k$-colorable.

To do this, first remove the vertex $v$ to produce a graph, $G_n$, with $n$ vertices. Let $u$ be a vertex that is adjacent to $v$ in $G_{n+1}$. Removing $v$ reduces the degree of $u$ by 1. So in $G_n$, vertex $u$ has degree strictly less than $k$. Since no edges were added, the vertex degrees of $G_n$ remain $\leq k$. So $G_n$ satisfies the conditions of the induction hypothesis, $P(n)$, and so we conclude that $G_n$ is $k$-colorable.

Now a $k$-coloring of $G_n$ gives a coloring of all the vertices of $G_{n+1}$, except for $v$. Since $v$ has degree less than $k$, there will be fewer than $k$ colors assigned to the nodes adjacent to $v$. So among the $k$ possible colors, there will be a color not used to color these adjacent nodes, and this color can be assigned to $v$ to form a $k$-coloring of $G_{n+1}$.

Solution. The flaw is that if $v$ has degree 0, then no such $u$ exists. In such a case, removing $v$ will not reduce the degree of any vertex, and so there may not be any vertex of degree less than $k$ in $G_n$, as in the counterexample of part (a).

So the mistaken sentence is “Let $u$ be a vertex that is adjacent to $v$ in $G_{n+1}$.”

Alternatively, you could say that it’s OK to reason about a nonexistent $u$, and the only mistake is the claim that $u$ exists. This claim is hidden in the phrase “So $G_n$ satisfies the conditions of the induction hypothesis, $P(n)$”.

(e) With a slightly strengthened condition, the preceding proof of the False Claim could be revised into a sound proof of the following Claim:

**Claim.** Let $G$ be a graph whose vertex degrees are all $\leq k$. If (statement inserted from below) has a vertex of degree strictly less than $k$, then $G$ is $k$-colorable.

Circle each of the statements below that could be inserted to make the proof correct.

- $G$ is connected and
- $G$ has no vertex of degree zero and
- $G$ does not contain a complete graph on $k$ vertices and
- every connected component of $G$
- some connected component of $G$

Solution. Either the first statement “$G$ is connected and” or the fourth statement “every connected component of $G$” will work.