Staff Solutions to In-Class Problems Week 6, Fri.

**STAFF NOTE:** Digraphs: Walks & Paths, Ch. 9–9.4

**Problem 1. (a)** Give an example of a digraph that has a closed walk including two vertices but has no cycle including those vertices.

**STAFF NOTE:** *Hint:* There is an example with 3 vertices.

**Solution.** Let the vertices be \(a, b, c\) and edges be \((a, b), (b, a), (b, c), (c, b)\). Now \(a\) and \(c\) are on the closed walk \(a, b, c, b, a\), but every closed walk from \(a\) to \(c\) must go through \(b\) at least twice, and so will not be a cycle.

**(b)** Prove Lemma 9.2.6:

**Lemma.** The shortest positive length closed walk through a vertex is a cycle.

**Solution.** *Proof.* Suppose \(w\) is a minimum positive length walk from \(u\) to \(u\). We claim \(w\) is a cycle.

To prove the claim, suppose to the contrary that \(w\) is not a cycle. One way the walk could fail to be a cycle is because some vertex besides \(u\) occurs twice on the walk:

**case** (some vertex \(x \neq u\) occurs twice in \(w\)): Then

\[ w = e \hat{x} f \hat{x} g \]

for some positive length walks \(e, f, g\). But then “deleting” \(f\) yields a strictly shorter walk, namely

\[ e \hat{x} g \]

is a shorter walk from \(u\) to \(u\), again contradicting the minimality of \(w\).

The other possibility is that no vertex besides \(u\) appears two or more times, but \(u\) appears more than two times:

**case** (\(u\) occurs more than two times in \(w\)): This means that

\[ w = e \hat{u} f \]

where both \(e\) and \(f\) have positive length. Then \(e\) is a shorter positive length walk from \(u\) to \(u\), contradicting the minimality of \(w\).

**Problem 2.**

A 3-bit string is a string made up of 3 characters, each a 0 or a 1. Suppose you’d like to write out, in one string, all eight of the 3-bit strings in any convenient order. For example, if you wrote out the 3-bit strings in the usual order starting with 000 001 010 . . . , you could concatenate them together to get a length \(3 \cdot 8 = 24\) string that started 000001010 . . .

But you can get a shorter string containing all eight 3-bit strings by starting with 00010 . . . . Now 000 is present as bits 1 through 3, and 001 is present as bits 2 through 4, and 010 is present as bits 3 through 5, . . .
(a) Say a string is \textit{3-good} if it contains every 3-bit string as 3 consecutive bits somewhere in it. Find a 3-good string of length 10, and explain why this is the minimum length for any string that is 3-good.

\textbf{Solution.} The string 0001110100 is a length 10 string that is 3-good. You can’t do better: there must be two bits to start and each additional bit can yield at most one new 3-bit string.

(b) Explain how any walk that includes every edge in the graph shown in Figure 1 determines a string that is 3-good. Find the walk in this graph that determines your 3-good string from part (a).

\textbf{Solution.} A string can be built up from any walk by starting with the \emph{bit-string} and successively adding the bit that labels the edge to the end of the string being built. If the walk includes every edge, then any string \( b_1 b_2 b_3 \) will appear as a substring when the edge \( b_1 b_2 \rightarrow b_2 b_3 \) appears in the walk.

In particular, the string 0001110100 is determined by the walk that goes through the following sequence of edges:

\[ (00 \rightarrow 00) \ (00 \rightarrow 01) \ (01 \rightarrow 11) \ (11 \rightarrow 11) \ (11 \rightarrow 10) \ (10 \rightarrow 01) \ (01 \rightarrow 10) \ (10 \rightarrow 00). \]

(c) Explain why a walk in the graph of Figure 1 that includes every edge \textit{exactly once} provides a minimum length 3-good string.

\textbf{Solution.} Since there are 8 edges, the string determined by the walk will be of length 10, which is the minimum possible as observed in part (a). Since the walk includes every edge, it will determine a 3-good string by part (b).

(d) The situation above generalizes to \( k \geq 2 \). Namely, there is a digraph, \( B_k \), such that \( V(B_k) := \{0, 1\}^k \), and any walk through \( B_k \) that contains every edge exactly once determines a minimum length \((k + 1)\)-good bit-string. What is this minimum length?

Define the transitions of \( B_k \). Verify that the in-degree and out-degree of every vertex is even, and that there is a positive path from any vertex to any other vertex (including itself) of length at most \( k \).

\textbf{Solution.} \( 2^{k+1} + k \).

A bit-string of length \( n \) has exactly \( n - k \) locations where a length-\((k + 1)\) subsequence can begin. Since there are \( 2^{k+1} \) length-\((k + 1)\) bit-strings, the minimum length, \( n \), of any \((k + 1)\)-good bit-string must satisfy \( n - k \geq 2^{k+1} \), so the minimum length is \( 2^{k+1} + k \). This is exactly the length of the bit-string that would be determined by a walk containing all \( 2 \cdot 2^k \) edges, \( E(B_k) \), in the graph \( B_k \).

\[ E(B_k) := \{\langle ax \rightarrow xb \rangle \mid x \in \{0, 1\}^{k-1} \text{ AND } a, b \in \{0, 1\}\} \]

If \( y \in \{0, 1\}^k \), then \( y = ax \) and \( y = zb \) for unique strings \( x, z \in \{0, 1\}^{k-1} \) and bits \( a, b \in \{0, 1\} \). Then by definition of \( E(B_k) \), there are exactly two edges out of \( y \), one going to \( x0 \) and the other to \( x1 \), so \( \text{outdeg}(y) = 2 \). Likewise, there are exactly two edges into \( y \), one from \( 0z \) and the other from \( 1z \), so \( \text{indeg}(y) = 2 \).

\textsuperscript{1}Problem 9.25 shows that if the in-degree of every vertex of a digraph is equal to its out-degree, and there are paths between any two vertices, then there is a closed walk that includes every edge exactly once. So the graph \( B_k \) implies that there always is a length-\(2^{k+1} + k\) bit-string in which every length-\((k + 1)\) bit-string appears as a substring. Such strings are known as \textit{de Bruijn sequences} having been studied by the great Dutch mathematician/logician Nicolaas de Bruijn, who died in February, 2012 at the age of 94.
To get from vertex $b_1 b_2 \ldots b_k$ to $c_1 c_2 \ldots c_k$ with a length-$k$ walk, proceed as follows:

$$b_1 b_2 b_3 \ldots b_k \rightarrow b_2 b_3 \ldots b_k c_1 \rightarrow b_3 \ldots b_k c_1 c_2 \rightarrow \cdots \rightarrow b_k c_1 c_2 \ldots c_{k-1} \rightarrow c_1 c_2 \ldots c_k.$$  

Since a walk of length $k$ exists, a path of length at most $k$ can be obtained by removing the cycles in the walk.

Supplemental Problem:

Problem 3.

Suppose that there are $n$ chickens in a farmyard. Chickens are rather aggressive birds that tend to establish dominance in relationships by pecking; hence the term “pecking order.” In particular, for each pair of distinct chickens, either the first pecks the second or the second pecks the first, but not both. We say that chicken $u$ virtually pecks chicken $v$ if either:

- Chicken $u$ directly pecks chicken $v$, or
- Chicken $u$ pecks some other chicken $w$ who in turn pecks chicken $v$.

A chicken that virtually pecks every other chicken is called a king chicken.

We can model this situation with a chicken digraph whose vertices are chickens with an edge from chicken $u$ to chicken $v$ precisely when $u$ pecks $v$. In the graph in Figure 2, three of the four chickens are kings. Chicken $c$ is not a king in this example since it does not peck chicken $b$ and it does not peck any chicken that pecks chicken $b$. Chicken $a$ is a king since it pecks chicken $d$, who in turn pecks chickens $b$ and $c$.

In general, a tournament digraph is a digraph with exactly one edge between each pair of distinct vertices.

(a) Define a 10-chicken graph with a king chicken that has outdegree 1.

Solution. 1 pecks 2 and 2 pecks 3–10 and 3–10 peck 1.
(b) Describe a 5-chicken graph in which every player is a king.

**Solution.** An example is illustrated in Figure 3.

(c) Prove Theorem (King Chicken Theorem). *The chicken with the largest outdegree in an n-chicken tournament is a king.*

**Solution. Proof.** By contradiction. Let \( u \) be a node in a tournament graph \( G = (V, E) \) with maximum outdegree and suppose that \( u \) is not a king. Let \( Y = \{v \mid (u \rightarrow v) \in E\} \) be the set of chickens that chicken \( u \) pecks. Then \( \text{outdeg}(u) = |Y| \).

Since \( u \) is not a king, there is a chicken \( x \not\in Y \) (that is, \( x \) is not pecked by chicken \( u \)) and that is not pecked by any chicken in \( Y \). Since for any pair of chickens, one pecks the other, this means that \( x \) pecks \( u \) as well as every chicken in \( Y \). This means that

\[
\text{outdeg}(x) = |Y| + 1 > \text{outdeg}(u).
\]

But \( u \) was assumed to be the node with the largest degree in the tournament, so we have a contradiction. Hence, \( u \) must be a king.

The King Chicken Theorem means that if the player with the most victories is defeated by another player \( x \), then at least he/she defeats some third player that defeats \( x \). In this sense, the player with the most victories has some sort of bragging rights over every other player. Unfortunately, as Figure 2 illustrates, there can be many other players with such bragging rights, even some with fewer victories.
STAFF NOTE: An extra supplemental problem if needed (unlikely):

Problem 4.
Lemma 9.2.5 states that dist \((u, v)\) \(\leq\) dist \((u, x)\) + dist \((x, v)\). It also states that equality holds iff \(x\) is on a shortest path from \(u\) to \(v\).

(a) Prove the “iff” statement from left to right.

Solution. Proof. To prove the “iff” from left to right, suppose dist \((u, v)\) = dist \((u, x)\) + dist \((x, v)\). Then merging a shortest path from \(u\) to \(x\) with shortest path from \(x\) to \(v\) yields a walk whose length is dist \((u, x)\) + dist \((x, v)\), which by assumption equals dist \((u, v)\). This walk must be a path or it could be shortened, giving a smaller distance from \(u\) to \(v\). So this is a shortest path containing \(x\).

(b) Prove the “iff” from right to left.

Solution. Proof. To prove the “iff” from right to left, suppose vertex \(x\) is on a shortest path \(w\) from \(u\) to \(v\), namely, \(w\) is a shortest path of the form \(f\hat{x}r\). The path \(f\) must be a shortest path from \(u\) to \(x\); otherwise replacing \(f\) by a shorter path from \(u\) to \(x\) would yield a shorter path from \(u\) to \(v\) than \(w\). Likewise \(r\) must be a shortest path from \(x\) to \(v\). So dist \((u, v)\) = \(|w| = |f| + |r| = \text{dist} \((u, x)\) + \text{dist} \((x, v)\).