In-Class Problems Week 7, Wed.

Problem 1.
Chickens are rather aggressive birds that tend to establish dominance over other chickens by pecking them—hence the term "pecking order." So for any two chickens in a farmyard, 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 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 1, 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.

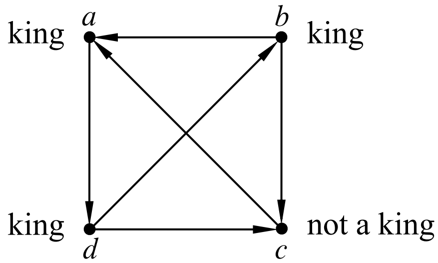

Figure 1 A 4-chicken tournament in which chickens a, b and d are kings.

.

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

(b) Describe a 5-chicken tournament graph in which every player is a king.

(c) Prove

Theorem (King Chicken Theorem)**.** *Any chicken with maximum out-degree in a tournament is 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 1 illustrates, there can be many other players with such bragging rights, even some with fewer victories.

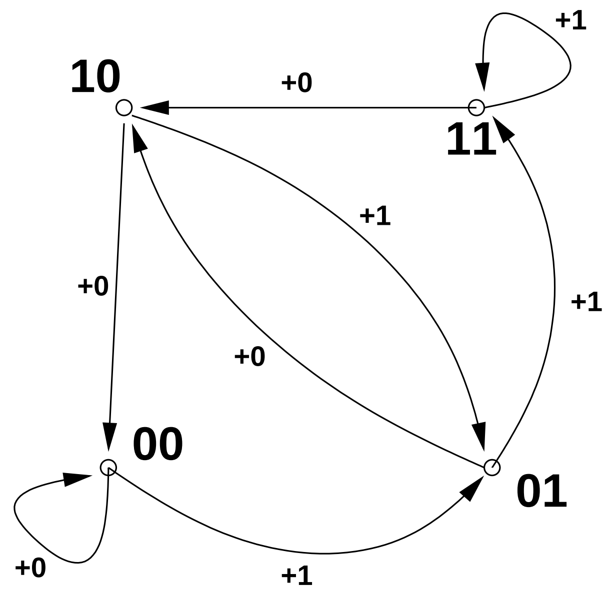

Figure 2 The 2-bit graph.

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 *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.

(b) Explain how any walk that includes every edge in the graph shown in Figure 2 determines a string that is 3-good. Find the walk in this graph that determines your 3-good string from part (a).

(c) Explain why a walk in the graph of Figure 2 that includes every every edge *exactly once* provides a minimum-length 3-good string.[1]

(d) Generalize the 2-bit graph to a k-bit digraph B_k for $k \geq 2$, where $V(B_k) ::= \{0,1\}^k$, and any walk through B_k that contains every edge exactly once determines a minimum length $(k+1)$-good bit-string.[2]

What is this minimum length?

Define the transitions of B_k. Verify that the in-degree of each vertex is the same as its out-degree and that there is a positive length path from any vertex to any other vertex (including itself) of length at most k.

Problem 3.
An *Euler tour*[3] of a graph is a closed walk that includes every edge exactly once. Such walks are named after the famous 17th century mathematician Leonhard Euler. (Same Euler as for the constant $e \approx 2.718$ and the totient function ϕ—he did a lot of stuff.)

[1] The 3-good strings explained here generalize to n-good strings for $n \geq 3$. They were studied by the great Dutch mathematician/logician Nicolaas de Bruijn, and are known as *de Bruijn sequences*. de Bruijn died in February, 2012 at the age of 94.

[2] Problem 10.7 explains why such "Eulerian" paths exist.

[3] In some other texts, this is called an *Euler circuit*.

So how do you tell in general whether a graph has an Euler tour? At first glance this may seem like a daunting problem (the similar sounding problem of finding a cycle that touches every vertex exactly once is one of those million dollar NP-complete problems known as the *Hamiltonian Cycle Problem*)—but it turns out to be easy.

(a) Show that if a graph has an Euler tour, then the in-degree of each vertex equals its out-degree.

A digraph is *weakly connected* if there is a "path" between any two vertices that may follow edges backwards or forwards.[4] In the remaining parts, we'll work out the converse. Suppose a graph is weakly connected, and the in-degree of every vertex equals its out-degree. We will show that the graph has an Euler tour.

A *trail* is a walk in which each edge occurs *at most* once.

(b) Suppose that a trail in a weakly connected graph does not include every edge. Explain why there must be an edge not on the trail that starts or ends at a vertex on the trail.

In the remaining parts, assume the graph is weakly connected, and the in-degree of every vertex equals its out-degree. Let **w** be the *longest* trail in the graph.

(c) Show that if **w** is closed, then it must be an Euler tour.

Hint: part (b)

(d) Explain why all the edges starting at the end of **w** must be on **w**.

(e) Show that if **w** was not closed, then the in-degree of the end would be bigger than its out-degree.

Hint: part (d)

(f) Conclude that if the in-degree of every vertex equals its out-degree in a finite, weakly connected digraph, then the digraph has an Euler tour.

---

[4]More precisely, a graph G is weakly connected iff there is a path from any vertex to any other vertex in the graph H with

$$V(H) = V(G), \text{ and}$$
$$E(H) = E(G) \cup \{\langle v \to u \rangle \mid \langle u \to v \rangle \in E(G)\}.$$

In other words $H = G \cup G^{-1}$.