CS70–Fall 2011 — Solutions to Homework 4

October 22, 2016

1. **Decomposing a non-Eulerian graph into Walks.**
   We split the $2d$ odd-degree vertices into d pairs, and join each pair with an edge, adding d more edges in total. Notice that now all vertices in this graph are of even degree. Now by Eulers theorem the resulting graph has an Eulerian tour. Removing the d added edges breaks the tour into d walks covering all the edges in the original graph, with each edge belonging to exactly one walk.

2. **Directing a graph.**

```
for each vertex v in the graph
   convert all the undirected edges from v to outward directed
       edges.
   if (!marked)
      marked = true;
   else
      remove this edge completely
```

   Suppose you start from a vertex A, and it has one edge to another vertex B. You convert this edge to a directed edge $A \to B$, but you do not remove this edge right now, because you also need $B \to A$. Therefore you mark it but setting the *marked* variable *true*. Whenever B's turn comes, it creates the edge from $B \to A$, sees that it has been marked, and thus removes it completely.

3. **Trees.**

   (a) Show that every tree contains at least one vertex of degree 1.
   Proof: by Contradiction
   Assume for the purpose of contradiction that every tree does not contain at least one vertex of degree 1. Now consider a very

simple tree with only 2 nodes. We know, from the definition of trees, that the tree has to be connected in such a way that there are no cycles. This means that both the vertices will have a single link between one another, and both will have degree 1 which contradicts our hypothesis. ✓

(b) Prove by induction on n that every tree with n vertices has exactly n 1 edges.

Proof: by Induction

Let n be the number of vertices in the tree, and $P(n)$ be the proposition that every tree with n vertices has exactly $n-1$ edges.

Base Case: With $n = 1$, there is only a single vertex, and because there can't be no loops, we have 0 edges. Thus $P(0)$ holds. ✓

Inductive step: Assume that $P(n)$ is true. In order to prove that $P(n+1)$ holds, consider a tree with n nodes. We already know that this tree has $n-1$ edges. Lets add another vertex to make it a new tree. According to the definition of tree, this new vertex cannot be isolated from the rest of tree, and it has to be minimally connected, which means that it will have only a single connection with one of the vertex already present in the tree. This will take our edge count to n. Thus $P(n+1)$ holds.

4. **Polygons.**

(a) $d(3) = 0$

(b) $d(4) = 2$

(c) Prove by Induction that $d(n) = \frac{n(n-3)}{2}$

Proof: by Induction

Given that

$$d(n) = \frac{n(n-3)}{2}$$

Base Case: When n = 3,

$$d(3) = 0 \checkmark$$

Inductive Step: Assume that $d(n)$ is true. We have to prove that

$$d(n+1) = \frac{(n+1)(n-2)}{2}$$

holds. Suppose that we have a polygon with n vertices. On adding the $n+1$ vertex to the polygon, the number of diagonals increase by $(n+1)-3$(you have to figure out why?), and obviously the vertex count increases by 1. So we have

$$d(n+1) = d(n) + 1 + ((n+1) - 3) = d(n) + n - 1$$

$$d(n+1) = \frac{n(n-3)}{2} + n - 1$$

$$d(n+1) = \frac{(n+1)(n-2)}{2} \checkmark$$

holds.

5. **Chains.**
   2^n. Consider a chain with $n = 1$. In this chain, we can only have 2 euler tours(from the starting node you can either go to the node at $+45°$ or the node at $-45°$), and adding each new link adds 2 more choices in the similar way. Note that at the bottom and top nodes of each link, you have no choice. You have to go to its right vertex no matter what.

6. **Hypercube routing.**

   (a) $1001 \to 0001 \to 0101 \to 0100$. I haven't done it here, but you people are encouraged to explore how the algorithm really works.

   (b) Each edge traversal corresponds to a separate bit position in which the two n-bit strings differ. Therefore, the number of traversals will be exactly the number of position at which the two strings differ. This can be given by the exclusive-OR of both n-bit strings, and counting the number of 1s in the resulting bit string.

   (c) From (b), we already know that the length of the path will be given by the number of positions in which the bit-string differ. Here, 010 and 100 differ in 2 bits and thus, the shortest path between them is 2. One possible path is $010 \to 110 \to 100$. The only other way to get from 010 to 100 in 2 steps is $010 \to 000 \to 100$, so $V(x, y) = \{010, 000, 110, 100\}$ and $E(x, y) = \{\{010, 000\}, \{010, 110\}, \{000, 100\}, \{110, 100\}\}$.

(d) Similarly to the above, we start by noting that no path from x to y can take fewer than $|x \oplus y|_1$ steps. There is always a path that takes exactly $|x \oplus y|_1$ steps as shown in part (b). Actually, take any string z that matches x at all the positions where y matches x (call these frozen bits), and has arbitrary bits in all the other positions. We want to show that $z \in V(x, y)$. Since bits are 0 or 1, each of the other positions must be 0 in one of x or y and 1 in the other one, so the non-frozen bits of z always match one of x or y. Take a path from x to y by (1) flipping all the non-frozen bits on which z matches y, and, then, (2) flipping all the non-frozen bits on which z matches x. Since we flip each non-frozen bit exactly once, this is still a shortest path (since its length is the number of non-frozen bits, $|x \oplus y|_1$). After finishing part (1) of this path, we must be exactly at z:

i. any frozen bits stay put throughout the entire path,
ii. the non-frozen bits on which z matches x havent been touched yet, and
iii. the remaining bits (all the non-frozen bits on which z matches y rather than x) just got flipped to match z.

Thus, z is on a shortest path from x to y, so $z \in V(x, y)$. Conversely, any w which does not match x and y on all the frozen bits cant be in $V(x, y)$: a path from x to y that goes through w must at least flip all the non-frozen bits once ($|x \oplus y|_1$ edges), and, for some frozen bit i on which w matches neither x nor y, that path must flip it at least twice (once on the way from x to w, and once on the way from w to y). Thus w cant be on a shortest path from x to y.

Any edges in the original hypercube that connect nodes z and $z0$ that are both in $V(x, y)$ must also be in $E(x, y)$, due to the following argument. Since the edge $\{z, z0\}$ is in the hypercube, z and $z0$ must differ by one non-frozen bit (they cant differ on a frozen bit, else they cant both be in $V(x, y)$). Without loss of generality, suppose that z matches x on this bit and $z0$ matches y (the only other possibility is z matching y and $z0$ matching x). Then, you can verify that bit-fixing from x to z, taking the $\{z, z0\}$ edge, and then bitfixing from $z0$ to y still yields a path of length $|x \oplus y|_1$ from x to y, making it a shortest path between x and y, and ensuring that $\{z, z0\} \in E$.