CS 70	Discrete Mathematics and Probability Theory	
Spring 2016	Rao and Walrand	Discussion 3A

1. Odd Degree Vertices

Claim: Let $G = (V, E)$ be an undirected graph. The number of vertices of G that have odd degree is even.

Prove the claim above using:

(i) Induction on $m = |E|$ (number of edges)

(ii) Induction on $n = |V|$ (number of vertices)

(iii) Well-ordering principle

(iv) Direct proof (e.g., counting the number of edges in G)

Answer: Let $V_{\text{odd}}(G)$ denote the set of vertices in G that have odd degree. We prove that $|V_{\text{odd}}(G)|$ is even.

(i) We use induction on $m \geq 0$.

Base case $m = 0$: If there are no edges in G, then all vertices have degree 0, so $V_{\text{odd}}(G) = \emptyset$.

Inductive hypothesis: Assume $|V_{\text{odd}}(G)|$ is even for all graphs G with m edges.

Inductive step: Let G be a graph with $m+1$ edges. Remove an arbitrary edge $\{u, v\}$ from G, so the resulting graph G' has m edges. By the inductive hypothesis, we know $|V_{\text{odd}}(G')|$ is even. Now add the edge $\{u, v\}$ to get back the original graph G. Note that u has one more edge in G than it does in G', so $u \in V_{\text{odd}}(G)$ if and only if $u \notin V_{\text{odd}}(G')$. Similarly, $v \in V_{\text{odd}}(G)$ if and only if $v \notin V_{\text{odd}}(G')$. The degrees of all other vertices are unchanged in going from G' to G. Therefore,

$$V_{\text{odd}}(G) = \begin{cases} V_{\text{odd}}(G') \cup \{u, v\} & \text{if } u, v \notin V_{\text{odd}}(G') \\ V_{\text{odd}}(G') \setminus \{u, v\} & \text{if } u, v \in V_{\text{odd}}(G') \\ (V_{\text{odd}}(G') \setminus \{u\}) \cup \{v\} & \text{if } u \in V_{\text{odd}}(G'), v \notin V_{\text{odd}}(G') \\ (V_{\text{odd}}(G') \setminus \{v\}) \cup \{u\} & \text{if } u \notin V_{\text{odd}}(G'), v \in V_{\text{odd}}(G') \end{cases}$$

so we see that $|V_{\text{odd}}(G)| - |V_{\text{odd}}(G')| \in \{-2, 0, 2\}$. Since $|V_{\text{odd}}(G')|$ is even, we conclude $|V_{\text{odd}}(G)|$ is also even.

(ii) We use induction on $n \geq 1$.

Base case $n = 1$: If G only has 1 vertex, then that vertex has degree 0, so $V_{\text{odd}}(G) = \emptyset$.

Inductive hypothesis: Assume $|V_{\text{odd}}(G)|$ is even for all graphs G with n vertices.

Inductive step: Let G be a graph with $n+1$ vertices. Remove a vertex v and all edges adjacent to it from G. The resulting graph G' has n vertices, so by the inductive hypothesis, $|V_{\text{odd}}(G')|$ is even. Now add the vertex v and all edges adjacent to it to get back the original graph G. Let $N_v \subseteq V$ denote the neighbors of v (i.e., all vertices adjacent to v). Among the neighbors N_v, the vertices in the intersection $A = N_v \cap V_{\text{odd}}(G')$ had odd degree in G', so they now have even degree in G. On the other hand, the vertices in $B = N_v \cap V_{\text{odd}}(G')^{\complement}$ had even degree in G', and they now have odd degree in G. The vertex v itself has degree $|N_v|$, so $v \in V_{\text{odd}}(G)$ if and only if $|N_v|$ is odd. We now consider two cases:

(a) Suppose $|N_v|$ is even, so $v \notin V_{\text{odd}}(G)$. Then

$$V_{\text{odd}}(G) = (V_{\text{odd}}(G') \setminus A) \cup B$$

so $|V_{\text{odd}}(G)| = |V_{\text{odd}}(G')| - |A| + |B|$. Note that A and B are disjoint and their union equals N_v, so $|A| + |B| = |N_v|$. Therefore, we can write $|V_{\text{odd}}(G)|$ as

$$|V_{\text{odd}}(G)| = |V_{\text{odd}}(G')| + |N_v| - 2|A|$$

which is even, since $|V_{\text{odd}}(G')|$ is even by the inductive hypothesis, and $|N_v|$ is even by assumption.

(b) Suppose $|N_v|$ is odd, so $v \in V_{\text{odd}}(G)$. Then

$$V_{\text{odd}}(G) = (V_{\text{odd}}(G') \setminus A) \cup B \cup \{v\}$$

so, again using the relation $|A| + |B| = |N_v|$, we can write

$$|V_{\text{odd}}(G)| = |V_{\text{odd}}(G')| - |A| + |B| + 1 = |V_{\text{odd}}(G')| + (|N_v| + 1) - 2|A|$$

which is even, since $|V_{\text{odd}}(G')|$ is even by the inductive hypothesis, and $|N_v|$ is odd by assumption.

This completes the inductive step and the proof.

Note how this proof is more complicated than the proof in part (i), even though they are both using induction. This tells you that choosing the right variable to induct on can simplify the proof.

(iii) Here we give a well-ordering proof using the number of edges m as the notion of "size" of G, so this is equivalent to the proof in part (i) using induction on m. (You can also try to give a well-ordering proof using n as the size of G.)

Suppose the contrary that the claim is false for some graphs. This means the set M is not empty, where M is the set of $m \in \mathbb{N}$ for which there exists a graph G with m edges that is a counterexample to the claim. Thus, we have a nonempty subset M of $\mathbb{N}$, so by the well-ordering principle, M has a smallest element m'. Note that $m' > 0$, since the claim is true for all graphs with 0 edges.

Let G be a graph with m' edges for which the claim is false, i.e., $|V_{\text{odd}}(G)|$ is odd (here we know such a G must exist from the definition of $m' \in M$). Remove one edge from G to obtain a smaller graph G' with $m' - 1$ edges (here we need $m' \geq 1$, which we have seen above). By our choice of m' as the smallest element of M, we know that $m' - 1 \notin M$, so the claim holds for G', namely, $|V_{\text{odd}}(G')|$ is even. Now add the removed edge to get back G. By the same argument as in the inductive step in part (i), this implies that $|V_{\text{odd}}(G)|$ is also even, a contradiction.

(iv) Let d_v denote the degree of vertex v (so $d_v = |N_v|$, where N_v is the set of neighbors of v). Observe that

$$\sum_{v \in V} d_v = 2m$$

because every edge is counted exactly twice when we sum the degrees of all the vertices. Now partition V into the odd degree vertices $V_{\text{odd}}(G)$ and the even degree vertices $V_{\text{odd}}(G)^{\complement}$, so we can write

$$\sum_{v \in V_{\text{odd}}(G)} d_v = 2m - \sum_{v \notin V_{\text{odd}}(G)} d_v.$$

Both terms in the righthand side above are even ($2m$ is even, and each term d_v is even because we are summing over even degree vertices $v \notin V_{\text{odd}}(G)$). So for the lefthand side $\sum_{v \in V_{\text{odd}}(G)} d_v$ to be even, we must have an even number of terms, since each term in the summation is odd. Therefore, there must be an even number of odd-degree vertices, namely, $|V_{\text{odd}}(G)|$ is even.

2. **Directed Euler**

Recall that in lecture we proved Euler's theorem for undirected graphs by giving a recursive algorithm whose correctness we proved by induction. One way to more fully understand the recursive algorithm is to unwind it into an iterative algorithm. In the case of Euler's theorem there is another reason to unwind the recursion — proving Euler's theorem for directed graphs via a recursive algorithm gets a little more tricky. In this exercise we will walk you through an iterative algorithm based proof of the *if* direction (the harder direction) of the directed Eulerian tour theorem. The theorem states that a directed graph G has an Eulerian tour that traverses each (directed) edge exactly once if and only if it is connected and for every vertex $v \in G$, $indegree(v) = outdegree(v)$. Here connected means that for any two edges in G it is possible to start from the first edge and walk along the directed graph edges to end up at the second edge.

(i) First prove that if you start from a vertex s and walk arbitrarily until you are stuck, the set of edges traversed forms a tour, T, (not necessarily Eulerian), and the remaining untraversed edges satisfy two properties:

  (a) For every vertex $v \in G$, $indegree(v) = outdegree(v)$.

  (b) There is some vertex v on T such that there is at least one untraversed edge (v, w) leaving v.

(ii) Fill in the details of an iterative procedure for finding the Eulerian tour, that in the next iteration starts from v and finds a tour T' among the untraversed edges. The iteration ends with an update to T by splicing T' into T.

(iii) Finally, prove by induction that your procedure results in an Eulerian tour.

Answer:

1. First we prove that you can only get stuck at vertex s. This is because we start with a graph G in which every vertex v has equal numbers of incoming and outgoing edges. Whenever the walk enters a vertex v it uses one incoming edge, and so the number of outgoing edges from v must be one more than the number of incoming edges, so the walk can always leave v. This shows that the walk can only get stuck at the original vertex s, and therefore, the walk defines a tour T.

   Let us now remove the edges in T from G, and call the resulting graph G'. We have to show that the graph G' satisfies properties (i) and (ii):

   i. Pick an arbitrary vertex v. Note that $indegree(v) = outdegree(v)$ in both G and T. Therefore when we remove all edges of T from G we are still left with $indegree(v) = outdegree(v)$ in G'.

   ii. We now show that if there are untraversed edges (i.e., there are some edges left in G'), then there is a vertex v on T such that there is at least one untraversed edge (v, w) leaving v (i.e., (v, w) is an edge in G'). Assume there is an untraversed edge (x, y) with x not on T. Since G is connected, there is a path starting at s that goes through the edge (x, y). Consider the first time this path traverses an edge (v, w) not on T. Clearly v is on T, since this is the first such edge, thus proving the claim.

2. Let us call the algorithm from part (a) FINDTOUR(G, s). Let SPLICE(T, T') denote the splicing operation to combine two edge-disjoint tours T and T' that share a vertex v. Specifically, SPLICE(T, T') returns a tour T'' obtained by traversing T from a vertex u until we reach v, then traversing the tour T' from v back to v again, and finally continue traversing T from v to the starting vertex u.

   The algorithm FINDEULERIANTOUR(G) is described below.

   function FINDEULERIANTOUR(G)

   pick an arbitrary vertex s and let $T = $ FINDTOUR(G, s)

```
remove edges in T from G
while there are untraversed edges:
    let v be a vertex on T such that there is at least one untraversed edge leaving v
    let T' = FINDTOUR(G, v)
    remove edges in T' from G
    let T = SPLICE(T, T')
return T
```

3. We claim that at the end of every iteration (where an iteration is defined to be the evaluation of the four lines in the while-loop in FINDEULERIANTOUR above) we maintain the invariant that: (i) T is a tour, (ii) the edges in T and G are disjoint, and (iii) T and G in total have m edges, where m is the original number of edges in G that we start with. To do this, we use induction on the number of iteration k.

   For the base case $k = 0$ (i.e., we have not done any while-loop iteration), we see that the first line $T = \text{FINDTOUR}(G,s)$ returns a tour. The second line removes edges in T from G, so this makes T and G edge-disjoint, and the total number of edges in T and G is equal to m, the original number of edges.

   Now suppose we have performed $k-1$ iterations of the while loop, for some $k \geq 1$. Consider the k-th iteration. (Here we assume there are still some untraversed edges, otherwise the algorithm will terminate after the $(k-1)$-st iteration.) We know from part (a) that there is always a vertex v on T such that there is at least one untraversed edge leaving v. Moreover, we also know that the remaining graph G' still satisfies the condition that every vertex has equal indegree and outdegree. Therefore, we can still call FINDTOUR on G to obtain a tour $T' = \text{FINDTOUR}(G,v)$. We splice T' into T, resulting in a larger tour $T = \text{SPLICE}(T,T')$. Since we remove edges in T' from G, the new tour T is still edge-disjoint from the new graph G, and the total number of edges in T and G is still equal to m. This completes the inductive step.

   Finally, observe that when the algorithm terminates, the graph G has no edges (since G only contains untraversed edges, and the algorithm terminates when there are no untraversed edges). Therefore, the claim above implies that all m edges are now contained in the tour T, and hence we conclude that T is an Eulerian tour.