CS 70 Spring 2017	Discrete Mathematics and Probability Theory Rao	HW 3

1 Sundry

Before you start your homework, write down your team. Who else did you work with on this homework? List names and email addresses. (In case of homework party, you can also just describe the group.) How did you work on this homework? Working in groups of 3-5 will earn credit for your "Sundry" grade.

Please copy the following statement and sign next to it:

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up. (Signature here)

2 Leaves in a Tree

A *leaf* in a tree is a vertex with degree 1.

(a) Prove that every tree on $n \geq 2$ vertices has at least two leaves.

(b) What is the maximum number of leaves in a tree with $n \geq 3$ vertices?

Solution:

(a) We give a direct proof. Consider the longest path $\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}$ between two vertices $x = v_0$ and $y = v_k$ in the tree (here the length of a path is how many edges it uses, and if there are multiple longest paths then we just pick one of them). We claim that x and y must be leaves. Suppose the contrary that x is not a leaf, so it has degree at least two. This means x is adjacent to another vertex z different from v_1. Observe that z cannot appear in the path from x to y that we are considering, for otherwise there would be a cycle in the tree. Therefore, we can add the edge $\{z, x\}$ to our path to obtain a longer path in the tree, contradicting our earlier choice of the longest path. Thus, we conclude that x is a leaf. By the same argument, we conclude y is also a leaf.

The case when a tree has only two leaves is called the *path graph*, which is the graph on $V = \{1, 2, \ldots, n\}$ with edges $E = \{\{1, 2\}, \{2, 3\}, \ldots, \{n-1, n\}\}$.

(b) We claim the maximum number of leaves is $n - 1$. This is achieved when there is one vertex that is connected to all other vertices (this is called the *star graph*).

We now show that a tree on $n \geq 3$ vertices cannot have n leaves. Suppose the contrary that there is a tree on $n \geq 3$ vertices such that all its n vertices are leaves. Pick an arbitrary vertex x, and let y be its unique neighbor. Since x and y both have degree 1, the vertices x, y form a connected component separate from the rest of the tree, contradicting the fact that a tree is connected.

3 Build-Up Error?

What is wrong with the following "proof"?

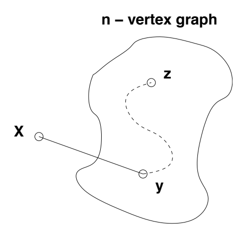
False Claim: If every vertex in an undirected graph has degree at least 1, then the graph is connected.

Proof: We use induction on the number of vertices $n \geq 1$.

Base case: There is only one graph with a single vertex and it has degree 0. Therefore, the base case is vacuously true, since the if-part is false.

Inductive hypothesis: Assume the claim is true for some $n \geq 1$.

Inductive step: We prove the claim is also true for $n+1$. Consider an undirected graph on n vertices in which every vertex has degree at least 1. By the inductive hypothesis, this graph is connected. Now add one more vertex x to obtain a graph on $(n+1)$ vertices, as shown below.



All that remains is to check that there is a path from x to every other vertex z. Since x has degree at least 1, there is an edge from x to some other vertex; call it y. Thus, we can obtain a path from x to z by adjoining the edge $\{x, y\}$ to the path from y to z. This proves the claim for $n+1$.

Solution:

The mistake is in the argument that *"every $(n+1)$-vertex graph with minimum degree 1 can be obtained from an n-vertex graph with minimum degree 1 by adding 1 more vertex."* Instead of starting by considering an arbitrary $(n+1)$-vertex graph, this proof only considers an $(n+1)$-vertex graph that you can make by starting with an n-vertex graph with minimum degree 1. As a counterexample, consider a graph on four vertices $V = \{1,2,3,4\}$ with two edges $E = \{\{1,2\},\{3,4\}\}$. Every vertex in this graph has degree 1, but there is no way to build this 4-vertex graph from a 3-vertex graph with minimum degree 1.

More generally, this is an example of *build-up error* in proof by induction. Usually this arises from a faulty assumption that every size $n+1$ graph with some property can be "built up" from a size n graph with the same property. (This assumption is correct for some properties, but incorrect for others, such as the one in the argument above.)

One way to avoid an accidental build-up error is to use a *"shrink down, grow back"* process in the inductive step: start with a size $n+1$ graph, remove a vertex (or edge), apply the inductive hypothesis $P(n)$ to the smaller graph, and then add back the vertex (or edge) and argue that $P(n+1)$ holds.

Let's see what would have happened if we'd tried to prove the claim above by this method. In the inductive step, we must show that $P(n)$ implies $P(n+1)$ for all $n \geq 1$. Consider an $(n+1)$- vertex graph G in which every vertex has degree at least 1. Remove an arbitrary vertex v, leaving an n-vertex graph G' in which every vertex has degree... uh-oh! The reduced graph G' might contain a vertex of degree 0, making the inductive hypothesis $P(n)$ inapplicable! We are stuck —– and properly so, since the claim is false!

4 Graph Coloring

Prove that a graph with maximum degree at most k is $(k+1)$-colorable.

Solution:

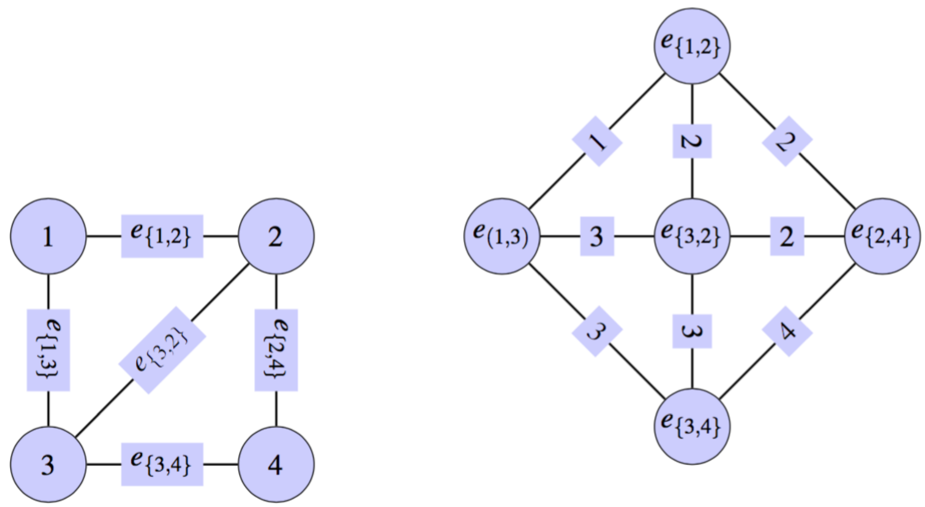
The natural way to try to prove this theorem is to use induction on k. Unfortunately, this approach leads to disaster. It is not that it is impossible, just that it is extremely painful and would ruin your week if you tried it on an exam. When you encounter such a disaster when using induction on graphs, it is usually best to change what you are inducting on. In graphs, typical good choices for the induction parameter are n, the number of nodes, or e, the number of edges.

We use induction on the number of vertices in the graph, which we denote by n. Let $P(n)$ be the proposition that an n-vertex graph with maximum degree at most k is $(k+1)$-colorable.

Base Case $n=1$: A 1-vertex graph has maximum degree 0 and is 1-colorable, so $P(1)$ is true.

Inductive Step: Now assume that $P(n)$ is true, and let G be an $(n+1)$-vertex graph with maximum degree at most k. Remove a vertex v (and all edges incident to it), leaving an n-vertex subgraph,H. The maximum degree of H is at most k, and so H is $(k+1)$-colorable by our assumption $P(n)$. Now add back vertex v. We can assign v a color (from the set of $k+1$ colors) that is different from all its adjacent vertices, since there are at most k vertices adjacent to v and so at least one of the $k+1$ colors is still available. Therefore, G is $(k+1)$-colorable. This completes the inductive step, and the theorem follows by induction.

5 Edge Complement



The **edge complement** graph of a graph $G = (V,E)$ is a graph $G' = (V',E')$, such that $V' = E$, and $(i, j) \in E'$ if and only if i and j had a common vertex in G. In the above picture, the graph on the right is the edge complement of the graph on the left: for every edge $e_{\{i,j\}}$ in the graph on the left there is a vertex in the graph on the right. If two edges $e_{\{i,j\}}$ and $e_{\{j,k\}}$ share a vertex j on the left, then the corresponding vertices on the right have an edge j connecting them.

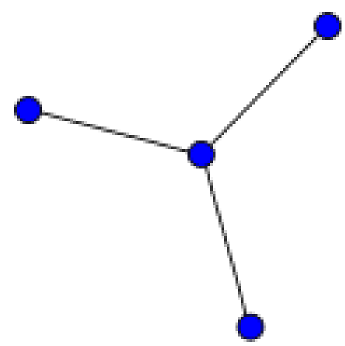
(a) Prove or disprove: if a graph G has an Eulerian tour, then its **edge complement** graph has an Eulerian tour.

(b) Prove or disprove: if a graph's **edge complement** graph G' has an Eulerian tour, then graph G has an Eulerian tour.

Solution:

(a) **True.** Using the same notation as above, we let an edge in G be $e_{\{i,j\}}$, with endpoints i and j. Then in G', $e_{\{i,j\}}$ is a vertex with edges between itself and only those vertices whose edge-representations in G shared the same vertex as itself. In other words, $e_{\{i,j\}}$ in G' will be neighbors with vertices of the form $e_{\{a,i\}}$ for some $a \neq i$ and $e_{\{j,b\}}$ for some $b \neq j$. If G had an Eulerian tour, then both i and j were incident to an even number of edges; this means that besides $e_{\{i,j\}}$, there were an odd number of other edges in G which were also incident to i, and likewise an odd number of other edges also incident to j. Thus in G', $e_{\{i,j\}}$ has an $odd + odd = even$ number of neighbors and thus is incident on an even number of edges. This is true for all vertices in G'; therefore, there is an Eulerian tour in G'.

(b) **False.** We will again use the same notation as above. If there is an Eulerian tour in G', any vertex $e_{\{i,j\}}$ will have an even number of neighbors. This means that in G, there are either an odd number of other edges besides $e_{\{i,j\}}$ incident to both i and j, or there are an even number of other edges incident to both i and j; i.e. in order for $e_{\{i,j\}}$ to in total have an even number of adjacent edges in G, the size of its two groups of neighbors must be either both an even number or both an odd number. If both groups of neighbors are odd, then both i and j have even degrees, since we add $e_{\{i,j\}}$ to the each group to make up the set of incident edges to i and j. However, if the number of neighbors in both groups are even, then both i and j will have odd degrees, since again we must add $e_{\{i,j\}}$ to both groups. If this is the case, then G will not have an Eulerian Tour. Also, a counterexample is a star with 3 vertices.



6 Proofs in Graphs

Please prove or disprove the following claims.

(a) Suppose we have n websites ($n \geq 2$) such that for every pair of websites A and B, either A has a link to B or B has a link to A. Prove or disprove that there exists a website that is reachable from every other website by clicking at most 2 links. (*Hint: Induction*)

(b) In the lecture, we have shown that a connected undirected graph has an Eulerian tour if and only if every vertex has even degree.

Prove or disprove that if a connected graph G on n vertices has exactly $2d$ vertices of odd degree, then there are d walks ($d > 0$) that *together* cover all the edges of G (i.e., each edge of G occurs in exactly one of the d walks; and each of the walks should not contain any particular edge more than once).

Solution:

(a) We prove this by induction on the number of websites n.

Base case For $n = 2$, there's always a link from one website to the other.

Induction Hypothesis When there are k websites, there exists a website w that is reachable from every other website by clicking at most 2 links.

Induction Step Let A be the set of websites with a link to w, and B be the set of websites two links away from w. The induction hypothesis states that the set of k websites $W = \{w\} \cup A \cup B$. Now suppose we add another website v. Between this website and every website in W, there must be a link from one to the other. If there is at least one link from v to $\{w\} \cup A$, w would still be reachable from v with at most 2 clicks. Otherwise, if all links from $\{w\} \cup A$ point to v, v will be reachable from every website in B with at most 2 clicks, because every website in B can click one link to go to a website in A, then click on one more link to go to v. In either case there exists a website in the new set of $k+1$ websites that is reachable from every other website by clicking at most 2 links.

(b) 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 Euler's 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.

7 Triangulated Planar Graph

In this problem you will prove that every triangulated planar graph (every face has 3 sides; that is, every face has three edges bordering it, including the unbounded face) contains either (1) a vertex of degree 1, 2, 3, 4, (2) two degree 5 vertices which are connected together, or (3) a degree 5 and a degree 6 vertices which are connected together. Justify your answers.

(a) Place a charge on each vertex v of value $6 - \text{degree}(v)$. What is the sum of the charges on all the vertices? (*Hint*: Use Euler's formula and the fact that the planar graph is triangulated.)

(b) What is the charge of a degree 5 vertex and of a degree 6 vertex?

(c) Move $1/5$ charge from each degree 5 vertex to each of its negatively charged neighbors. Conclude the proof in the case where there is a degree 5 vertex with positive remaining charge.

(d) If no degree 5 vertices have positive charge after discharging, does there exist a vertex with positive charge after discharging? If there is such a vertex, what are possible degrees of that vertex?

(e) Suppose there exists a degree 7 vertex with positive charge after the discharging process of degree 5 vertices. How many neighbors of degree 5 might it have?

(f) Continuing the last question. Since the graph is triangulated, are two of these degree 5 vertices adjacent?

(g) Finish the proof from the facts you obtained from the previous questions.

Solution:

(a) Let V be the vertex set , E be the edge set, F be the faces in the graph, we have

$$\begin{aligned} \sum_{v \in V} 6 - \text{degree}(v) &= 6|V| - \sum_{v \in V} \text{degree}(v) && (1) \\ &= 6|V| - 2|E|. && (2) \end{aligned}$$

The last step is because that we count each edge twice as degree for each end vertex. And since the graph is triangulated, each face uses exactly three edges and each edge is shared by two faces, so we can substitute $|F| = 2|E|/3$ in Euler's formula to get

$$\begin{aligned} |V| + |F| &= |E| + 2 && (3) \\ |V| + \frac{2|E|}{3} &= |E| + 2 && (4) \\ 3|V| + 2|E| &= 3|E| + 6 && (5) \\ |E| &= 3|V| - 6. && (6) \end{aligned}$$

Substitute (6) into (2) to get that the sum of charge is 12.

(b) The charge is 1 for degree 5 vertex, and 0 for degree 6 vertex.

(c) If there is a degree 5 vertex with positive remaining charge, that means at least one of its neighbors is not negatively charged. In other words, at least one of its neighbors has degree 1, 2, 3, 4, 5, or 6, which proves the statement.

(d) Yes, there exists a vertex with positive charge. Since we know that the sum of charge of the entire graph is 12, it is impossible to have no positively charged vertex.

The possible degrees of vertex that have positive charge after discharging are $1, 2, 3, 4, 7$. Vertices with degree greater than 8 will have initial charge -2, and will not have positive charge even if all their neighbors are of degree 5.

(e) At least 6 out of the 7 are degree 5.

(f) Since the graph is triangulated, observe that fixing a drawing of the planar graph, we can order neighbors of the degree 7 node clockwise. And every two consecutive neighbors (defined by the ordering) form a triangle with the degree 7 vertex. From Part (e) we know that at least 6 out of the 7 are degree 5. Therefore, it is impossible that none of these degree 5 vertices are connected to another degree 5 node.

(g) We split the proof into several cases. When there is a degree 5 vertex with positive remaining charge, the statement is true by Part (c). When there is no degree 5 vertex with positive remaining charge, we know from Part (d) that either there is a positive charged vertex with degree $1, 2, 3, 4$ or with degree 7. In the later case, we know that at least two degree 5 vertices are adjacent from Part (f) which concludes our proof.

8 Hypercube Routing

Recall that an n-dimensional hypercube contains 2^n vertices, each labeled with a distinct n bit string, and two vertices are adjacent if and only if their bit strings differ in exactly one position.

(a) The hypercube is a popular architecture for parallel computation. Let each vertex of the hypercube represent a processor and each edge represent a communication link. Suppose we want to send a packet from vertex x to vertex y. Consider the following "bit-fixing" algorithm:

> In each step, the current processor compares its address to the destination address of the packet. Let's say that the two addresses match up to the first k positions (reading the bits from left to right). The processor then forwards the packet and the destination address on to its neighboring processor whose address matches the destination address in at least the first $k+1$ positions. This process continues until the packet arrives at its destination.

Consider the following example where $n = 4$: Suppose that the source vertex is (1001) and the destination vertex is (0100). Give the sequence of processors that the packet is forwarded to using the bit-fixing algorithm.

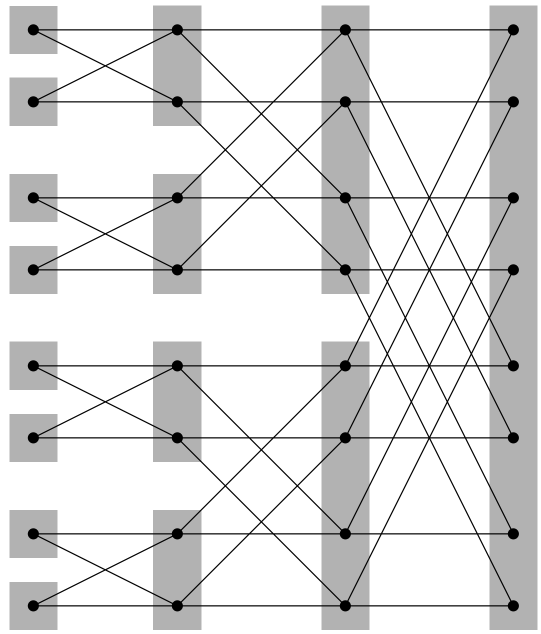
(b) The *Hamming distance* $H(x,y)$ between two n-bit strings x and y is the number of bit positions where they differ. Show that for an arbitrary source vertex and arbitrary destination vertex, the number of edges that the packet must traverse under this algorithm is the Hamming distance between the n-bit strings labeling source and destination vertices.

(c) Consider the following example where $n = 3$: Suppose that x is (110) and y is (011). What is the length of the shortest path between x and y? What is the set of all vertices and the set of all edges that lie on shortest paths between x and y? Do you see a pattern? You do not need to prove your answer here – you'll provide a general proof in part (d).

(d) Answer the last question for an arbitrary pair of vertices x and y in the hypercube. Can you describe the set of vertices and the set of edges that lie on shortest paths between x and y? Prove that your answers are correct. (*Hint:* Consider the bits where x and y differ.)

(e) There is another famous graph, called the butterfly network, which is another popular architecture for parallel computation. You will see this network in CS 170 in the context of circuits for implementing the FFT (fast fourier transform). Here is a diagram of the butterfly network for $n = 3$. In general, the butterfly network has $(n+1)\cdot 2^n$ vertices organized into $n+1$ columns of 2^n vertices each. The vertices in each column are labeled with the bit strings in $\{0,1\}^n$, and all vertices in the same row have the same label. The source is on the leftmost column and the destination is on the right.

It turns out the n-butterfly network is equivalent to the n-dimensional hypercube unrolled into n bit-fixing steps. On the graph below, trace all the paths from source $x = (110)$ to destination $y = (011)$, so that these paths are the shortest bit-fixing paths that you obtained from part (c). For this, you need to label the vertices in the graph explicitly. This example should convince you that the butterfly network is indeed equivalent to the hypercube routing.



Solution:

(a) The source $x = (1001)$ and $y = (0100)$ differ in three bits, and the bit-fixing algorithm sequentially flips the differing bits from left to right, so the sequence of processors from x to y is:

$$1001 \to 0001 \to 0101 \to 0100.$$

(b) We proceed by induction on $k = H(x,y)$. Let $P(k)$ be the proposition that k is the number of edges that the packet must traverse under this algorithm if the Hamming distance between strings x and y is k.

Base Case $k = 0$*:* The strings x and y are identical. Thus 0 edge is needed.

Inductive Step: For $H(x,y) = k+1$, after the first step, the bit-fixing algorithm sends the packet from x to a neighboring vertex x' which is one step closer to y. i.e. $H(x',y) = H(x,y) - 1 = (k+1) - 1 = k$. Now by the induction hypothesis, the packet must traverse $H(x',y) = k$ edges to go from x' to y. Thus, the packet must traverse $1+k$ edges from x to y.

(c) The length of the shortest path is 2, and we see there are two paths:

$$110 \to 010 \to 011 \qquad \text{and} \qquad 110 \to 111 \to 011.$$

Note that the first path is the same path obtained by the bit-fixing algorithm from part (a), where we flip the bits from left to right, while the second path is flipping the bits from right to left. Therefore, the set of all vertices in the shortest paths is

$$\{110, 010, 111, 011\}.$$

Here the pattern is that we look at the starting and ending vertices $x = (110)$ and to $y = (011)$, and hold the middle bit where they agree constant, and try all possible patterns for the first and last bits where they disagree. Another way of saying this is that the set of vertices is the 2 dimensional sub cube with middle bit equal to 1.

(d) By part (b), the length of the shortest path between x and y is the Hamming distance $k = H(x,y)$. To achieve this shortest path, we must leave alone all bit positions where x and y agree, and in each step change one of the bit positions where x and y disagree (to look like y). This means that the set of all vertices in shortest paths corresponds exactly to the 2^k n-bit strings which agree with x and y where the two are the same, and have an arbitrary set of k bits in those positions where x and y disagree. In other words, this set of vertices forms a k-dimensional subcube of the n-dimensional hypercube, and all edges in this subcube appear in the shortest paths.

(e) In this problem, we asked you to label the vertices such that:

- The vertices in each column are labeled with the 2^n bit strings $\{0,1\}^n$.
- The vertices in each row all have the same label.
- The unique path from every source (on the left) to every destination (on the right) is the same path obtained by running the bit-fixing algorithm from part (a). In particular, flipping the leftmost bit corresponds to moving from the first column to the second, then flipping the second bit corresponds to moving from the second column to the third, and so on. In each step, a horizontal edge represents we keep that bit fixed, while a diagonal edge means we flip that bit.

The figure below shows a possible labeling for $n = 3$. Once we fix the label of one vertex, then the connection of the diagram uniquely determines the labels of the other vertices. For example, if we want the top left vertex to be 000, then the vertex below that must be 100, since the first step of the bit-fixing algorithm sends 000 to either 000 itself (the horizontal line), or to 100 (the diagonal line). The red line in the figure below traces the path from 110 to 011 via 010, which is the same path from the bit-fixing algorithm.