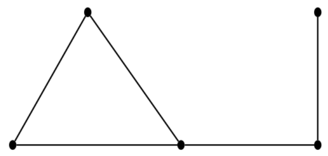
CS 70 Fall 2019 — Discrete Mathematics and Probability Theory — Alistair Sinclair and Yun S. Song — HW 3

Note: *This homework consists of two parts. The first part (questions 1-6) will be graded and will determine your score for this homework. The second part (questions 7-9) will be graded if you submit them, but will not affect your homework score in any way. You are strongly advised to attempt all the questions in the first part. You should attempt the problems in the second part only if you are interested and have time to spare.*

Part 1: Required Problems

1 Degree Sequences

The *degree sequence* of a graph is the sequence of the degrees of the vertices, arranged in descending order, with repetitions as needed. For example, the degree sequence of the following graph is $(3,2,2,2,1)$.



For each of the parts below, determine if there exists a simple undirected graph G (i.e. a graph without self-loops and multiple-edges) having the given degree sequence. Justify your claim.

(a) $(3,3,2,2)$

(b) $(3,2,2,2,2,1,1)$

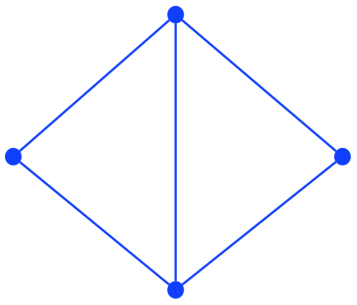
(c) $(6,2,2,2)$

(d) $(4,4,3,2,1)$

Solution:

(a) **Yes**

The following graph has degree sequence $(3,3,2,2)$.



(b) **No**

For any graph G, the number of vertices that have odd degree is even. The given degree sequence has 3 odd degree vertices.

(c) **No**

The total number of vertices is 4. Hence there cannot be a vertex with degree 6.

(d) **No**

The total number of vertices is 5. Hence, any degree 4 vertex must have an edge with every other vertex. Since there are two degree 4 vertices, there cannot be a vertex with degree 1.

2 Bipartite Graphs

An undirected graph is bipartite if its vertices can be partitioned into two disjoint sets L, R such that each edge connects a vertex in L to a vertex in R (so there does not exist an edge that connects two vertices in L or two vertices in R).

(a) Suppose that a graph G is bipartite, with L and R being a bipartite partition of the vertices. Prove that $\sum_{v \in L} \deg(v) = \sum_{v \in R} \deg(v)$.

(b) Suppose that a graph G is bipartite, with L and R being a bipartite partition of the vertices. Let s and t denote the average degree of vertices in L and R respectively. Prove that $s/t = |R|/|L|$.

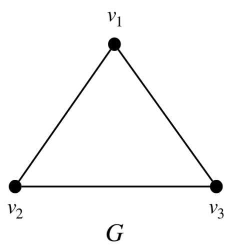
(c) The *double* of a graph G consists of two copies of G with edges joining the corresponding "mirror" vertices. More precisely, if $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices and E the set of edges, then the double of the graph G is given by $G_1 = (V_1, E_1)$, where

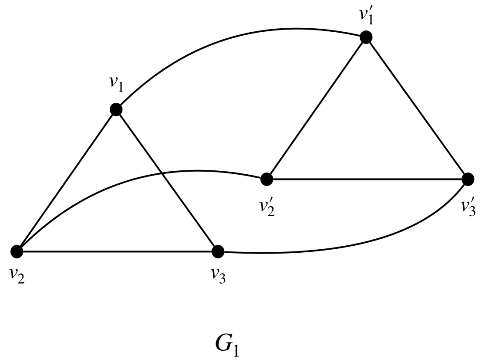
$$V_1 = \{v_1, v_2, \ldots, v_n, v'_1, v'_2, \ldots, v'_n\},$$

and

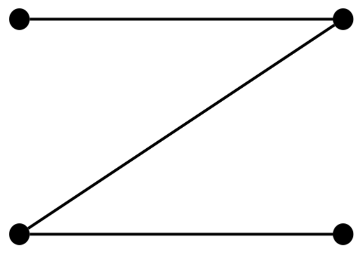
$$E_1 = E \cup \{(v'_i, v'_j) | (v_i, v_j) \in E\} \cup \{(v_i, v'_i), \forall i\}.$$

Here is an example,





Draw the double of the following graph:



(d) Now suppose that G_1 is a bipartite graph, G_2 is the double of G_1, G_3 is the double of G_2, and so on. (Each G_{i+1} has twice as many vertices as G_i). Show that $\forall n \geq 1$, G_n is bipartite.

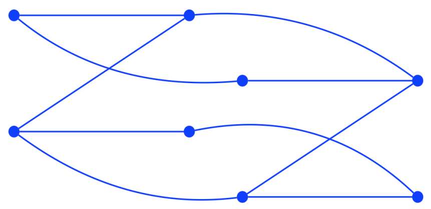
Hint: Use induction on n.

Solution:

(a) Since G is bipartite, each edge connects one vertex in L with a vertex in R. Since each edge contributes equally to $\sum_{v \in L} \deg(v)$ and $\sum_{v \in R} \deg(v)$, we see that these two values must be equal.

(b) By part (a), we know that $\sum_{v \in L} \deg(v) = \sum_{v \in R} \deg(v)$. Thus $|L| \cdot s = |R| \cdot t$. A little algebra gives us the desired result.

(c) The double of the graph in part (c) is as shown below:



(d) We use induction. Let $P(n)$ be the proposition that G_n is bipartite. The base case is when $n = 1$. We see that $P(1)$ must be true since G_1 is bipartite by assumption. Now suppose that for $k \geq 1$, $P(k)$ holds. We see that the graph G_{k+1} consists of two subgraphs, each having the same structure as G_k, except the edges joining the corresponding vertices of the two subgraphs. Ignore these extra edges, i.e, the edges joining the corresponding vertices of the two subgraphs. Since $P(k)$ is true, we can label the two subgraphs into disjoint sets $\{L_1, R_1\}$ and $\{L_2, R_2\}$. Then we can define new sets $L = \{L_1, R_2\}$ and $R = \{R_1, L_2\}$ that are disjoint. Every edge connects a vertex from L_1 to R_1 and from L_2 to R_2, so it connects from L to R. Now considering the extra edges that we ignored, we see that each such edge connects a vertex from L_1 to a vertex in L_2 and a vertex in R_1 to a vertex in R_2. Hence, every edge connects from L to R. Thus, the graph G_{k+1} is bipartite.

3 Planarity and Graph Complements

Let $G = (V, E)$ be an undirected graph. We define the complement of G as $\overline{G} = (V, \overline{E})$ where $\overline{E} = \{(i, j) | i, j \in V, i \neq j\} - E$; that is, $\overline{G}$ has the same set of vertices as G, but an edge e exists is $\overline{G}$ if and only if it does not exist in G.

(a) Suppose G has v vertices and e edges. How many edges does $\overline{G}$ have?

(b) Prove that for any graph with at least 13 vertices, G being planar implies that $\overline{G}$ is non-planar.

(c) Now consider the converse of the previous part, i.e., for any graph G with at least 13 vertices, if $\overline{G}$ is non-planar, then G is planar. Construct a counterexample to show that the converse does not hold.

Hint: Recall that if a graph contains a copy of K_5, then it is non-planar. Can this fact be used to construct a counterexample?

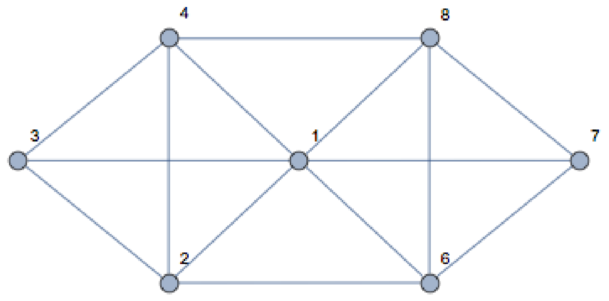
Solution:

(a) If G has v vertices, then there are a total of $\frac{v(v-1)}{2}$ edges that could possibly exist in the graph. Since e of them appear in G, we know that the remaining $\frac{v(v-1)}{2} - e$ must appear in $\overline{G}$.

(b) Since G is planar, we know that $e \leq 3v - 6$. Plugging this in to the answer from the previous part, we have that $\overline{G}$ has at least $\frac{v(v-1)}{2} - (3v - 6)$ edges. Since v is at least 13, we have that $\frac{v(v-1)}{2} \geq \frac{v \cdot 12}{2} = 6v$, so $\overline{G}$ has at least $6v - 3v + 6 = 3v + 6$ edges. Since this is strictly more than the $3v - 6$ edges allowed in a planar graph, we have that $\overline{G}$ must not be planar.

(c) The converse is not necessarily true. As a counterexample, suppose that G has exactly thirteen vertices, of which five are all connected to each other and the remaining eight have no edges incident to them. This means that G is non-planar, since it contains a copy of K_5. However, $\overline{G}$ also contains a copy of K_5 (take any 5 of the 8 vertices that were isolated in G), so $\overline{G}$ is also non-planar. Thus, it is possible for both G and $\overline{G}$ to be non-planar.

4 Eulerian Tour and Eulerian Walk



(a) Is there an Eulerian tour in the graph above? If no, give justification. If yes, provide a counterexample.

(b) Is there an Eulerian walk in the graph above? If no, give justification. If yes, provide a counterexample.

(c) What is the condition that there is an Eulerian walk in an undirected graph? Briefly justify your answer.

Solution:

(a) No. Two vertices have odd degree.

(b) Yes. One of the two vertices with odd degree must be the starting vertex, and the other one must be the ending vertex. For example: $3,4,2,1,3,2,6,1,4,8,1,7,8,6,7$ will be an Eulerian walk (the numbers are the vertices visited in order). Note that there are 14 edges in the graph.

(c) An undirected graph has an Eulerian walk if and only if it is connected (except for isolated vertices) and has exactly two odd degree vertices.

Justifications: *Only if.* Suppose there exists an Eulerian walk, say starting at u and ending at v (note that u and v are distinct, otherwise it will be a tour and not a walk). Then all the vertices that lie on this walk are connected to each other and all the vertices that do not lie on this walk (if any) must be isolated. Thus the graph is connected (except for isolated vertices). Moreover, every intermediate visit to a vertex in this walk is being paired with two edges, and therefore, except for u and v, all other vertices must be of even degree.

If. For a connected graph with no odd degree vertices, we have shown in the lectures that there is an Eulerian tour.

Solution 1: Suppose G is connected (except for isolated vertices) and has exactly two odd degree vertices, say u and v. First remove the isolated vertices if any. Since u and v belong to a connected component, one can find a path from u to v. Consider the graph obtained by removing the edges of the path from the graph. In the resulting graph, all the vertices have even degree. Hence, for each connected component of the residual graph, we find an Eulerian tour. (Note that the graph obtained by removing the edges of the path can be disconnected.) Observe that an Eulerian walk is simply an edge-disjoint walk that covers all the edges. What we just did is decomposing all the edges into a path from u to v and a bunch of edge-disjoint Eulerian tours. A path is clearly an edge-disjoint walk. Then, given an edge-disjoint walk and an edge-disjoint tour such that they share at least one common vertex, one can combine them into an edge-disjoint walk simply by augmenting the walk with the tour at the common vertex. Therefore we can combine all the edge-disjoint Eulerian tours into the path from u to v to make up an Eulerian walk from u to v.

Solution 2: Alternatively, take the two odd degree vertices u and v, and add a vertex w with two edges (u,w) and (w,v). The resulting graph G' has only vertices of even degree (we added one to the degree of u and v and introduced a vertex of degree 2) and is still connected. So, we can find an Eulerian tour on G'. Now, delete the component of the tour that uses edges (u,w) and (w,v). The part of the tour that is left is now an Eulerian walk from u to v on the original graph, since it traverses every edge on the original graph.

5 Trees and Components

(a) Bob removed a degree 3 node from an n-vertex tree. How many connected components are there in the resulting graph? Please provide an explanation.

(b) Given an n-vertex tree, Bob added 10 edges to it and then Alice removed 5 edges. If the resulting graph has 3 connected components, how many edges must be removed in order to remove all cycles from the resulting graph? Please provide an explanation.

Solution:

(a) **3.**

Let the original graph be denoted by $G = (V, E)$ and the resulting graph after Bob removes the node be denoted by $G' = (V', E')$. Let $|V| = n$ and hence $|E| = n - 1$ by the Theorem proved in class. Also, $|V'| = n - 1$ and $|E'| = n - 4$. Let k denote the number of connected components in G'. Since removing vertices and edges should not give rise to cycles, we know that the graph G' is acyclic. Hence each of the connected components is a tree. Let $n_1, n_2, \ldots, n_k$ denote the number of nodes in each of the k connected components respectively. Again by the Theorem proved in class, we have that each of the component consists of $n_1 - 1, n_2 - 1, \ldots, n_k - 1$ edges respectively. Thus the total number of edges in G' is

$$n-4 = |E'| = \sum_{i=1}^{k}(n_i - 1) = (\sum_{i=1}^{k} n_i) - k = (n-1) - k.$$

Hence $k = 3$.

Alternate Solution. Here we use the fact that removing an edge from a forest (i.e an acyclic graph) increases the number of components by exactly 1. If we remove the three edges incident on the vertex removed by Bob, we get 4 components. However, Bob has also removed the degree 3 vertex which itself is one of the four connected components. Hence we are left with 3 connected components.

(b) **7.**

We first note that in any connected graph if we remove an edge belonging to a cycle, then the resulting graph is still connected. Hence for any connected graph, we can repeatedly remove edges belonging to cycles, until no more cycles remain. This process will give rise to a connected acyclic graph, i.e., a tree.

Since the final graph we wish to obtain is acyclic, each of its connected component must be a tree. Thus the components should have $n_1 - 1$, $n_2 - 1$ and $n_3 - 1$ edges each, where n_1, n_2, n_3 are the number of vertices in each of these components. Let n denote the total number of vertices and hence $n = n_1 + n_2 + n_3$. As a result, the total number of edges in the final graph is $n - 3$. The total number of edges after Bob and Alice did their work was $n - 1 + 10 - 5 = n + 4$. Thus one needs to remove 7 edges.

6 Hamiltonian decompositions of a complete graph

(a) (Walecki's Theorem) Let K_n be the complete graph on n vertices.

(i) If n is an odd positive integer, then the edge set of K_n can be partitioned into x edge-disjoint Hamiltonian cycles (a Hamiltonian cycle is a cycle where each vertex appears exactly once).

(ii) If n is an even positive integer, then the edge set of K_n can be partitioned into y edge-disjoint Hamiltonian cycles and one perfect matching (a perfect matching is a set of edges such that every vertex is a part of exactly one edge in the set).

Assume the above theorem is true. What are x and y?

(b) Give a partition of the edges of K_5 with vertices $\{1,2,3,4,5\}$ into edge-disjoint Hamiltonian cycles.

(c) Give a partition of the edges of K_6 with vertices $\{1,2,3,4,5,6\}$ into edge-disjoint Hamiltonian cycles and a perfect matching.

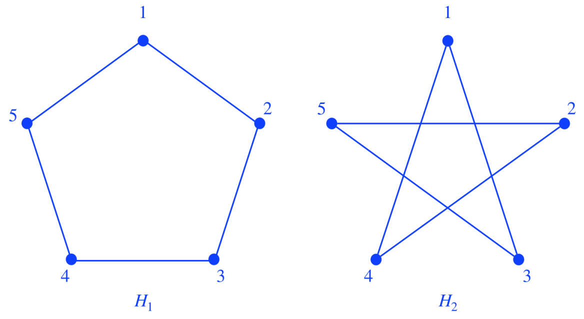
Solution:

(a) $\mathbf{x = (n-1)/2, y = (n-2)/2}$.

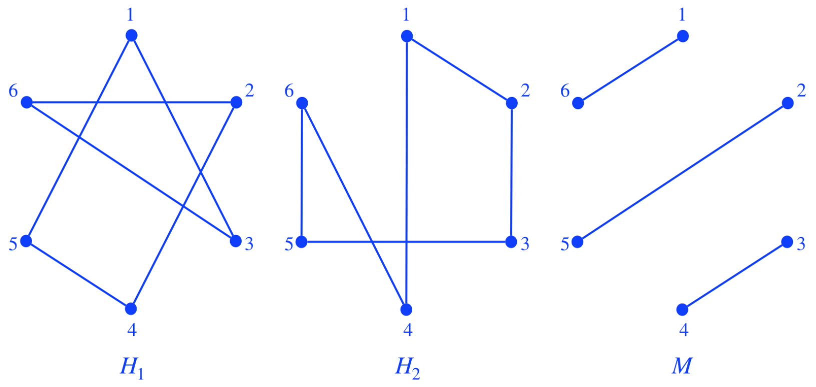
(i) Suppose n is odd. The total number of edges in K_n is $n(n-1)/2$. Each Hamiltonian cycle consists of n edges. Hence, the number of Hamiltonian cycles $x = (n-1)/2$.

(ii) Suppose n is even. The total number of edges in K_n is $n(n-1)/2$. A perfect matching consists of $n/2$ edges. Hence, the number of Hamiltonian cycles $y = [n(n-1)/2 - n/2]/n = (n-2)/2$.

(b) Here is one possible answer.
$H_1 = (\{1,2\}), \{2,3\}, \{3,4\}, \{4,5\}, \{5,1\})$
$H_2 = (\{1,3\}, \{3,5\}, \{5,2\}, \{2,4)\}, \{4,1\})$



(c) Here is one possible answer.
$H_1 = (\{1,3\}), \{3,6\}, \{6,2\}, \{2,4\}, \{4,5\}, \{5,1\})$
$H_2 = (\{1,4\}, \{4,6\}, \{6,5\}, \{5,3)\}, \{3,2\}, \{2,1\})$
$M = (\{1,6\}, \{2,5\}, \{3,4\})$



Note: *This concludes the first part of the homework. The problems below are optional, will not affect your score, and should be attempted only if you have time to spare.*

Part 2: Optional Problems

7 Planarity

Consider graphs with the property T: For every three distinct vertices v_1, v_2, v_3 of graph G, there are at least two edges among them. Prove that if G is a graph on ≥ 7 vertices, and G has property T, then G is nonplanar.

Solution:

In this problem, we use proof by contradiction. Assume G is planar. Select any five vertices out of the seven. Consider the subgraph formed by these five vertices. They cannot form K_5, since G is planar. So some pair of vertices amongst these five has no edge between them. Label these vertices v_1 and v_2. The remaining five vertices of G besides v_1 and v_2 cannot form K_5 either, so there is a second pair of vertices amongst these new five that has no edge between them. Label these v_3 and v_4. Label the remaining three vertices v_5, v_6 nd v_7. Since $v_1 v_2$ is not an edge, by property T it must be that $\{v_1, v\}$ and $\{v_2, v\}$ are edges, where $v \in \{v_3, v_4, v_5, v_6, v_7\}$. Similarly for v_3, v_4 we have that $\{v_3, v\}$ and $\{v_4, v\}$ are edges, where $v \in \{v_1, v_2, v_5, v_6, v_7\}$. Now consider the subgraph induced by $\{v_1, v_2, v_3, v_5, v_6, v_7\}$. With the three vertices $\{v_1, v_2, v_3\}$ on one side and $\{v_5, v_6, v_7\}$ on the other, we observe that $K_{3,3}$ is a subgraph of this induced graph. This contradicts the fact that G is planar.

The above shows that any graph with 7 vertices and property T is non-planar. Any graph with greater than 7 vertices and property T will also be non-planar because it will contain a subgraph with 7 vertices and property T.

8 Connectivity

Consider the following claims regarding connectivity:

(a) Prove: If G is a graph with n vertices such that for any two non-adjacent vertices u and v, it holds that $\deg u + \deg v \geq n-1$, then G is connected.

[*Hint:* Show something more specific: for any two non-adjacent vertices u and v, there must be a vertex w such that u and v are both adjacent to w.]

(b) Give an example to show that if the condition $\deg u + \deg v \geq n-1$ is replaced with $\deg u + \deg v \geq n-2$, then G is not necessarily connected.

(c) Prove: For a graph G with n vertices, if the degree of each vertex is at least $n/2$, then G is connected.

(d) Prove: If there are exactly two vertices with odd degrees in a graph, then they must be in the same connected component (meaning, there is a path connecting these two vertices).

[*Hint:* Proof by contradiction.]

Solution:

(a) If u and v are two adjacent vertices, they are connected by definition. Then, consider non-adjacent u and v. Then, there must be a vertex w such that u and v are both adjacent to w. To see why, suppose this is not the case. Then, the set of neighbors of u and v has $n-1$ elements, but there are only $n-2$ other vertices. (This is the Pigeonhole Principle.) We have proven that for any non-adjacent u and v, there is a path $u \to w \to v$, and thus G is connected.

(b) Consider the graph formed by two disconnected copies of K_2. For non-adjacent u, v, it holds that $\deg u + \deg v = 2 = 4 - 2 = n - 2$, but the graph is not connected.

(c) If each vertex's degree is at least $n/2$, then for any two non-adjacent vertices u, v,

$$\deg u + \deg v \geq \frac{n}{2} + \frac{n}{2} = n > n - 1.$$

Then by part (a), the graph is connected.

(d) Suppose that they are not connected to each other. Then they must belong to two different connected components, say G_1 and G_2. Each of them will only have one vertex with odd degree. This leads to a contradiction since the sum of all degrees should be an even number.

9 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 "left-to-right bit-fixing" algorithm:

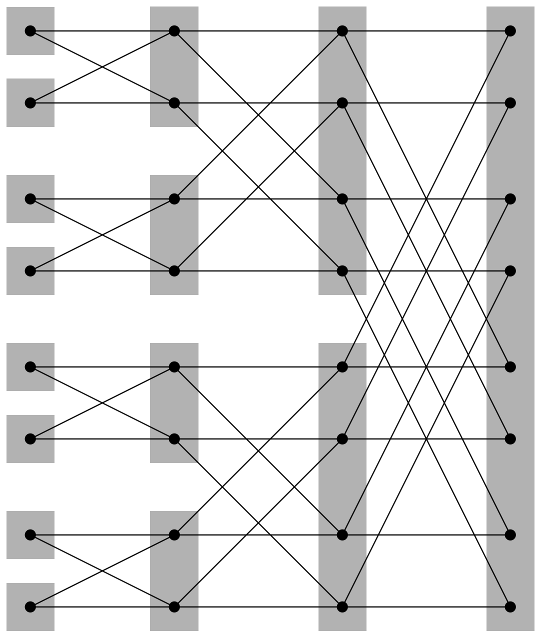
> In each step, the current processor compares its address to the destination address of the packet. Let's say that the two addresses match up to the first k positions (reading the bits from left to right). The processor then forwards the packet and the destination address on to its neighboring processor 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 left-to-right 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 the left-to-right bit-fixing algorithm is the Hamming distance between the n-bit strings labeling source and destination vertices.

(c) 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 left-to-right bit-fixing steps. On the graph below, label the vertices in the graph explicitly and trace the path from source $x = (110)$ to destination $y = (011)$. Convince yourself that the left to right paths in the butterfly network are indeed equivalent to the hypercube routing obtained by the left-to-right bit-fixing algorithm.



Solution:

(a) The source $x = (1001)$ and $y = (0100)$ differ in three bits, and the left-to-right 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.

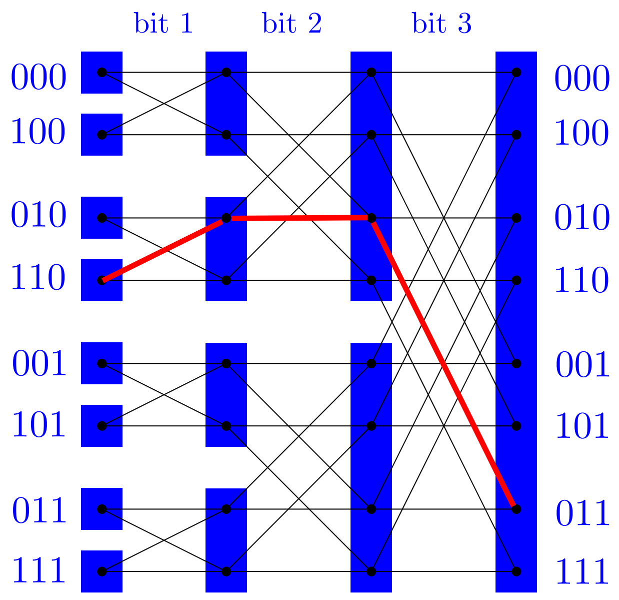
Base Case $k = 0$: The strings x and y are identical. Thus 0 edge is needed.

Inductive Step: For $H(x,y) = k+1$, after the first step, the left-to-right 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) 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 left-to-right 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 left-to-right bit-fixing algorithm.


bit 1
bit 2
bit 3
000
100
010
110
001
101
011
111
000
100
010
110
001
101
011
111