CS 70 Discrete Mathematics and Probability Theory
Spring 2015 Vazirani HW 4

Reminder: Deadline for HW3 self-grade is Thursday, February 12 at noon.

Reading: Note 5 and Note 6.

Due Monday February 16

1. **Trees**

   Recall that a **tree** is a connected acyclic graph. Note that this is similar to but slightly different from a rooted tree that you might have come across before in the context of data structures. In our context the tree does not have any designated root vertex, and is just a graph on n vertices and with m edges. We start by proving two useful properties of trees:

   (a) Any pair of vertices in a tree are connected by exactly one (simple) path.

   (b) Adding any edge to a tree creates a simple cycle.

   Now show that if a graph satisfies either of these two properties then it must be a tree:

   (c) If for every pair of vertices in a graph they are connected by exactly one simple path, then the graph must be a tree.

   (d) If the graph has no simple cycles but has the property that the addition of any single edge (not already in the graph) will create a simple cycle, then the graph is a tree.

   Answer:

   (a) Pick any pair of vertices x, y. We know there is a path between them since the graph is connected. We will prove that this path is unique by contradiction:

   Suppose there are two distinct paths from x to y. At some point (say at vertex a) the paths must diverge, and at some point (say at vertex b) they must reconnect. So by following the first path from a to b and the second path in reverse from b to a we get a cycle. This gives the necessary contradiction.

   (b) Pick any pair of vertices x, y not connected by an edge. We prove that adding the edge $\{x,y\}$ will create a simple cycle. From part (a), we know that there is a unique path between x and y. Therefore, adding the edge $\{x,y\}$ creates a simple cycle obtained by following the path from x to y, then following the edge $\{x,y\}$ from y back to x.

   (c) Assume we have a graph with the property that there is a unique simple path between every pair of vertices. We will show that the graph is a tree, namely, it is connected and acyclic. First, the graph is connected because every pair of vertices is connected by a path. Moreover, the graph is acyclic because there is a unique path between every pair of vertices. More explicitly, if the graph has a cycle, then for any two vertices x, y in the cycle there are at least two simple paths between them (obtained by going from x to y through the right or left half of the cycle), contradicting the uniqueness of the path. Therefore, we conclude the graph is a tree.

(d) Assume we have a graph with no simple cycles, but adding any edge will create a simple cycle. We will show that the graph is a tree. We know the graph is acyclic because it has no simple cycles. To show the graph is connected, we prove that any pair of vertices x, y are connected by a path. We consider two cases: If $\{x,y\}$ is an edge, then clearly there is a path from x to y. Otherwise, if $\{x,y\}$ is not an edge, then by assumption, adding the edge $\{x,y\}$ will create a simple cycle. This means there is a simple path from x to y obtained by removing the edge $\{x,y\}$ from this cycle. Therefore, we conclude the graph is a tree.

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.

(a) 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:

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

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

(b) 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.

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

Answer:

(a) 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.

(b) 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 $T$
```

(c) 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 $T=$ 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 $T'=$ FINDTOUR(G,v). We splice T' into T, resulting in a larger tour $T=$ 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.

3. **Hamiltonian path**

A *Hamiltonian path* in an undirected graph $G=(V,E)$ is a path that goes through every vertex *exactly once*. A *Hamiltonian cycle* (or *Hamiltonian tour*) is a cycle that goes through every vertex exactly once. Note that, in a graph with n vertices, a Hamiltonian path consists of $n-1$ edges, and a Hamiltonian cycle consists of n edges.

Prove that for every $n \geq 2$, the n-dimensional hypercube has a Hamiltonian cycle.

Answer:

We proceed by induction on n. In the base case $n = 2$, we have the 2-dimensional hypercube, which is a square graph on $V = \{00, 01, 10, 11\}$. Here we have a Hamiltonian cycle $00 \to 01 \to 11 \to 10 \to 00$.

Suppose now that the $(n-1)$-dimensional hypercube has a Hamiltonian cycle. Let $v \in \{0,1\}^{n-1}$ be a vertex adjacent to 0^{n-1} (the notation 0^{n-1} means a sequence of $n-1$ zeroes) in the Hamiltonian cycle. By removing the edge $\{0^{n-1}, v\}$ from the cycle, we obtain a Hamiltonian path in the $(n-1)$-dimensional hypercube that starts at 0^{n-1} and ends at v.

We now want to construct a Hamiltonian cycle in the n-dimensional hypercube. Recall the decomposition of the n-dimensional hypercubes into 0-subcube and 1-subcube, where the 0-subcube (respectively, the 1-subcube) is the $(n-1)$-dimensional hypercube with vertices labeled by $0x$ (respectively, $1x$) for $x \in \{0,1\}^{n-1}$, and every vertex $0x$ in the 0-subcube is connected to the corresponding vertex $1x$ in the 1-subcube.

Then the following is a Hamiltonian cycle in an n-dimensional hypercube: have a path that goes from $0^n \in \{0,1\}^n$ to $0v$ by passing through all vertices in the 0-subcube (this is simply a copy of the Hamiltonian path in dimension $(n-1)$ from 0^{n-1} to v), then an edge from $0v$ to $1v$, then a path from $1v$ to 10^{n-1} that passes through all vertices in the 1-subcube (this is another copy of the Hamiltonian path in dimension $(n-1)$ traveled in reverse), and finally an edge from 10^{n-1} to 0^n. This completes the proof.

4. **Tournament**

A *tournament* is defined to be a directed graph such that for every pair of distinct nodes v and w, exactly one of (v,w) and (w,v) is an edge (representing which player beat the other in a round-robin tournament). Prove that every tournament has a Hamiltonian path. In other words, you can always arrange the players in a line so that each player beats the next player in the line.

Answer: We provide two possible answers: one using simple induction, and the other using strong induction.

Answer 1: We will prove this with induction on the number of nodes/players.

Base Case $n = 1$: There is only one player, so the claim is trivially true.

Inductive Hypothesis: Suppose for some $n \geq 1$, we can find a Hamiltonian path in a tournament of n players.

Inductive Step: Consider a tournament of $n+1$ players. Arbitrarily pick one player p_{n+1} to "hold out." From our inductive hypothesis, we can arrange the remaining n players in a line, say $p_1, p_2, \ldots, p_n$, such that p_i beat p_{i+1} for $1 \leq i \leq n-1$.

Let p_a be the last player that beat p_{n+1}. If there is no such p_a (i.e., p_{n+1} beat everyone), then we can place p_{n+1} before p_1, and we are done. Otherwise, reorder the players as follows:

$$p_1, p_2, \ldots, p_a, p_{n+1}, p_{a+1}, \ldots, p_n.$$

We know that p_{n+1} must have beaten p_{a+1} by definition (or else p_{a+1} would be the last player that beat p_{n+1}). If it turns out that $a = n$, we simply place p_{n+1} after p_n, and we still have a valid Hamiltonian path.

Therefore, for all $n \geq 1$, there exists a Hamiltonian path in a tournament of n players. $\square$

Answer 2: We will prove this with strong induction on the number of nodes/players.

Base Case $n = 1$: There is only one player, so the claim is trivially true.

Inductive Hypothesis: Suppose for all $1 \leq k < n$, we can find a Hamiltonian path in a tournament of k players.

Inductive Step: Consider a tournament of n players.

Arbitrarily pick one player p to "hold out." Let S be the set of players who beat p, and let T be the set of players who p beat. From our inductive hypothesis, we can find a Hamiltonian path in S and in T. Finally, to obtain a Hamiltonian path on all n players, we connect the last person in S to p, and p to the first person in T.

Therefore, for all $n \geq 1$, there exists a Hamiltonian path in a tournament of n players. □

5. **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 for 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. 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.

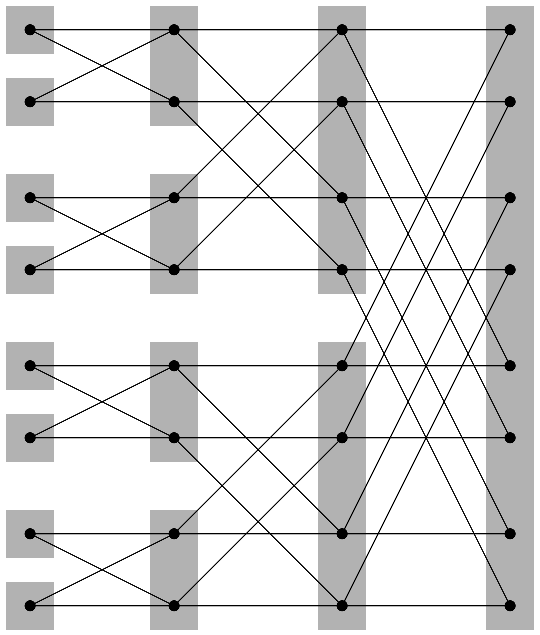
(c) Consider the following example where $n = 3$: Suppose that x is (110) and y is (011). What is the length of the shortest path between x and y? What is the set of all vertices and the set of all edges that lie on shortest paths between x and y? Do you see a pattern? You do not need to prove your answer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



Answer:

(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) This is an easy induction, since 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. $H(x', y) = H(x, y) - 1$. Now by the induction hypothesis (you have to spell out what it is!), the packet must traverse $H(x', y) = H(x, y) - 1$ edges to go from x' to y, for a grand total of $H(x', y) + 1 = H(x, y)$.

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

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

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

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

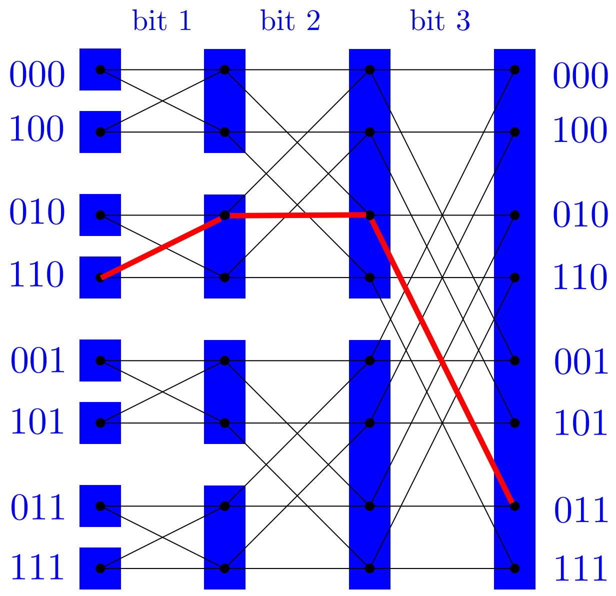
Here the pattern is that we look at the starting and ending vertices $x = (110)$ and to $y = (011)$, and hold the middle bit where they agree constant, and try all possible patterns for the first and last bits where they disagree. Another way of saying this is that the set of vertices is the 2 dimensional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



6. **Modular arithmetic**

Solve the following equations (i.e., find all solutions) for x and y modulo the indicated modulus, or show that no solution exists. Show your work.

(a) $8x \equiv 1 \pmod{15}$.

(b) $6x + 14 \equiv 11 \pmod{23}$.

(c) $5x + 13 \equiv 4 \pmod{20}$.

(d) $5x + 14 \equiv 4 \pmod{20}$.

(e) The system of simultaneous equations $2x + 3y \equiv 0 \pmod 7$ and $3x + y \equiv 4 \pmod 7$.

Answer: Recall that the multiplicative inverse of y modulo n, if it exists, is defined to be the unique number y^{-1} satisfying $y \cdot y^{-1} \equiv 1 \pmod n$. Also recall from Theorem 6.2 in Note 6 that the multiplicative inverse $y^{-1} \pmod n$ exists if and only if $\gcd(y,n) = 1$.

In particular, for any integers y,z, we see the following: If $\gcd(y,n) = 1$, then the equation $x \cdot y \equiv z \pmod n$ has a unique solution $x \equiv y^{-1} \cdot z \pmod n$. On the other hand, if $\gcd(y,n) \neq 1$, then $y^{-1} \pmod n$ does not exist, but the equation $x \cdot y \equiv z \pmod n$ may or may not have solutions in this case.

(a) Since $\gcd(8,15) = 1$, we know 8 has a multiplicative inverse modulo 15. Note that $8 \times 2 \equiv 16 \equiv 1 \pmod{15}$, so 2 is the multiplicative inverse of 8 modulo 15. Therefore, the unique solution to the equation $8x \equiv 1 \pmod{15}$ is

$$x \equiv 8^{-1} \equiv 2 \pmod{15}.$$

(b) We simplify $6x + 14 \equiv 11 \pmod{23}$ into $6x \equiv -3 \pmod{23}$, which has a unique solution $x \equiv -3 \times 6^{-1} \pmod{23}$. Here we know 6 has a multiplicative inverse modulo 23 because $\gcd(6,23) = 1$. Note that $6 \times 4 \equiv 24 \equiv 1 \pmod{23}$, so 4 is the multiplicative inverse of 6 modulo 23. Therefore, the solution is

$$x \equiv -3 \times 6^{-1} \equiv -3 \times 4 \equiv -12 \equiv 11 \pmod{23}.$$

(c) We simplify $5x + 13 \equiv 4 \pmod{20}$ into $5x \equiv -9 \pmod{20}$. Note that $\gcd(5,20) = 5 \neq 1$, so 5 does not have a multiplicative inverse modulo 20. This by itself, however, does not imply that the equation $5x \equiv -9 \pmod{20}$ has no solution. To do that, we argue as follows: For any integer x, the quantity $5x$ can only be equal to 0, 5, 10, or 15 modulo 20. Therefore, the equation $5x \equiv -9 \pmod{20}$ has no solution.

(d) We simplify $5x + 14 \equiv 4 \pmod{20}$ into $5x \equiv -10 \pmod{20}$. Here we are in the same situation as in part (c) where $\gcd(5,20) = 5 \neq 1$, so 5 does not have a multiplicative inverse modulo 20. However, in this case the equation $5x \equiv -10 \pmod{20}$ has solutions.

More precisely, note that the modular equation $5x \equiv -10 \pmod{20}$ is equivalent to the (non-modular) equation $5x = 20y - 10$ for some integer y. Simplifying, we get $x = 4y - 2$. So we see that any integer x of the form $x = 4y - 2$ for some integer y is a solution to the equation $5x \equiv -10 \pmod{20}$, which we can immediately verify:

$$5x \equiv 5(4y-2) \equiv 20y - 10 \equiv -10 \pmod{20}.$$

In particular, this shows that there are infinitely many integer solutions x, one for every choice of y. However, since we want to find the solutions modulo 20, it suffices to consider the values $y \in \{0,1,2,3,4\}$, in which case we get $-2, 2, 6, 10, 14$ as the possible values for $x \equiv 4y - 2 \pmod{20}$. Therefore, the solutions to the equation $5x \equiv -10 \pmod{20}$ are:

$$x \equiv 2, 6, 10, 14, 18 \pmod{20}.$$

(e) We multiply the first equation by $2^{-1} \equiv 4 \pmod 7$:

$$2^{-1} \times (2x+3y) \equiv 2^{-1} \times 0 \pmod 7 \quad \Rightarrow \quad x+5y \equiv 0 \pmod 7$$

Similarly, we multiply the second equation by $3^{-1} \equiv 5 \pmod 7$:

$$3^{-1} \times (3x+y) \equiv 3^{-1} \times 4 \pmod 7 \quad \Rightarrow \quad x+5y \equiv 6 \bmod 7$$

However, the resulting two equations above contradict each other. Therefore, no solutions exist.

7. **Extra Credit**

Alice and Bob have two positive integers, x and y respectively, glued to their foreheads, so that each can read the other's number but not their own. They also know that $|x-y| = 1$. The following conversation is overheard between Alice and Bob:

(1) Alice: I don't know my number.
(2) Bob: I don't know my number.
(3) Alice: I don't know my number.
(4) Bob: I don't know my number.
...
They say this to each other 200 times. And then we overhear them say:

(201) Alice: I know my number!
(202) Bob: I know my number too!

Can you explain the conversation and figure out the numbers x and y? You may assume that Alice and Bob each know that the other is brilliant— so each is confident that if the other has sufficient information to deduce the answer then he/she definitely will.

Answer: We provide two possible solutions.

Answer 1: Define a *round* of the conversation to be a sequence of Alice saying "I don't know my number" and then Bob saying "I don't know my number."

Let us consider what happens in the first round. When Alice says "I don't know my number", Bob concludes that $y > 1$. This is because if $y = 1$, then Alice would have concluded that $x = 2$ (since $|x-y| = 1$, so $x = 0$ or $x = 2$, but we also know x is a positive integer, so x must be equal to 2). Next, when Bob says "I don't know my number", Alice concludes that $x > 2$. This is because if $x = 1$ then Bob knows $y = 2$; similarly, if $x = 2$ then Bob knows $y = 3$ (since y cannot be equal to 1.) Therefore, after the first round, both Alice and Bob know that $x > 2$ and $y > 1$.

We generalize this and claim that at the end of the n-th round, both Alice and Bob know that $x > 2n$ and $y > 2n-1$. We prove this using induction on n. The preceding paragraph shows the base case $n = 1$ is true. Suppose the claim is true for $n-1$; we also want to prove it is true for n. Suppose now that the conversation runs for at least n rounds. After the $(n-1)$-th round, by the inductive hypothesis both Alice and Bob know that $x > 2n-2$ and $y > 2n-3$. Now consider what happens at the n-th round. When Alice says "I don't know my number", Bob concludes that $y > 2n-1$. This is because if $y = 2n-2$ then Alice would have concluded that $x = 2n-1$ (since x can't be $2n-3$), and if $y = 2n-1$ then Alice would have concluded that $x = 2n$ (since x can't be $2n-2$). By the same argument, when Bob says "I don't know my number", Alice concludes that $x > 2n$. Therefore, at the end of the n-th round, both Alice and Bob know that $x > 2n$ and $y > 2n-1$. This completes the inductive step.

We now claim that if the conversation ends after n rounds (i.e., at the $(n+1)$-st round Alice says "I know my number!" and Bob says "I know my number too!"), then either $(x,y) = (2n+1, 2n)$ or

$(x,y) = (2n+2, 2n+1)$. This is because from the claim above, after n rounds both Alice and Bob know that $x > 2n$ and $y > 2n-1$. If $y = 2n$ then Alice concludes $x = 2n+1$, and if $y = 2n+1$ then Alice concludes $x = 2n+2$ (since x can't be $2n$). On the other hand, if $y > 2n+1$, then Alice still does not know whether her number x is equal to $y+1$ or $y-1$, since both options are still valid ($> 2n$). Therefore, the only way for Alice to conclude "I know my number!" is if $y = 2n$ or $y = 2n+1$. Bob therefore makes the same deduction that there are two possible cases, and because Bob can see Alice's number x, he knows which case he is in, and hence he also announces "I know my number too!"

Finally, the original problem is the special case $n = 100$, in which case we conclude Alice's number is $x = 201$ and Bob's number is $y = 200$, or Alice's number is $x = 202$ and Bob's number is $y = 201$. □

Remark: The proof above still leaves the big question: since Alice and Bob each sees at the beginning of the game that the other number is bigger than 200, each of them knows that for the first 100 rounds each will say "I don't know my number." Why then does it help them to hear the other say it in each round? The key is to understand how common knowledge propagates between Alice and Bob. In the beginning Alice and Bob each has their own knowledge (of each other's number), and the conversation helps them share the information with each other. The second answer below shows more explicitly how common knowledge between Alice and Bob propagates one day at a time.

Answer 2: We say that a fact is *common knowledge* if everyone knows it and additionally everyone knows that it is common knowledge (thus everyone knows that everyone knows it, and everyone knows *that*, and so on). We assume that the problem statement is common knowledge, and that when someone says something it becomes common knowledge that they said it.

We'll prove that after "I don't know my number" has been said k times, it is common knowledge that the next player to speak has a number which is strictly greater than k. (Note that here k refers to the number of statements, not rounds as in the previous answer, where a round consists of 2 statements.)

For $k = 0$ this is trivial, because it is common knowledge that the numbers are positive. So consider the $(k+1)$-st time that a player says "I don't know my number." Suppose that Alice is the player speaking. Before Alice spoke, by induction it was common knowledge that her number was more than k. Thus it was common knowledge that Bob's number is more than $k-1$. If Bob's number was n, then Alice would know that her number is $k+1$ (since it can't be $k-1$). Similarly, if Bob's number was $k+1$, then Alice would know that her number is $k+2$ (since it can't be k). So when Alice says "I don't know my number" it implies that Bob's number can't be k or $k+1$. Moreover, this argument relies on facts that are common knowledge, so both players can carry it out and the conclusion becomes common knowledge. So after Alice's statement it is common knowledge that Bob's number is strictly more than $k+1$, as desired.

Now suppose that Alice and Bob each says "I don't know my number" 100 times, as stated in the problem. It is then common knowledge that Alice's number is at least 201 and Bob's number is at least 200. If Bob's number is exactly 200, then Alice now knows her number to be 201 (since it can't be 199). If Bob's number is 201, then Alice now knows her number to be 202 (since it can't be 200). If Bob's number is 202 or greater, then Alice would not have been able to deduce her number.

Therefore, since Alice declares she knows her number in the 201-st statement, we conclude that either Alice's number is 201 and Bob's number is 200, or Alice's number is 202 and Bob's number is 201. Moreover, after Alice reveals her number, Bob can deduce these two possibilities. By looking at Alice's number he knows which case he is in, and thus he also declares that he knows his number.

So we have two possibilities: $(x,y) = (201,200)$ or $(x,y) = (202,201)$. □

8. **Extra Credit**

Let G be an undirected graph on $n \geq 3$ vertices, and assume that all vertices in G are of degree at least $n/2$. Prove that G has a Hamiltonian cycle.

Answer: We use proof by contradiction. Assume that G doesn't have a Hamiltonian cycle. Add new edges to G as long as we can without creating a Hamiltonian cycle. When we stop, we have a graph G' in which all n vertices have degree at least $n/2$, there is no Hamiltonian cycle, but adding any new edge would create a Hamiltonian cycle.

Let x and y be two vertices in G' that are not connected by an edge. As adding the edge $\{x,y\}$ would create a Hamiltonian cycle, it follows that G' has a Hamiltonian path P that starts at x and ends at y. Let $x = z_1, z_2, \ldots, z_n = y$ denote the vertices of this path (which are also all the vertices in the graph, since the path is Hamiltonian).

Let N_x denote the set of indices $2 \leq j \leq n$ such that z_j is a neighbor of x, i.e., $\{x, z_j\}$ is an edge. Similarly, let M_y denote the set of indices $2 \leq j \leq n$ such that z_{j-1} is a neighbor of y. We claim that there is an index $i \in \{2, \ldots, n\}$ such that $i \in N_x \cap M_y$. To see this, note that we have two subsets N_x and M_y of $\{2, 3, \ldots, n\}$, each of which has size at least $n/2$ by assumption, so they must have a nonempty intersection (for otherwise, if $N_x \cap M_y = \emptyset$, then $n \leq |N_x| + |M_y| = |N_x \cup M_y| \leq |\{2, 3, \ldots, n-1\}| = n-1$, a contradiction).

Therefore, we can find an index $i \in N_x \cap M_y$. This means $i \in N_x$, so $\{x, z_i\}$ is an edge, and $i-1 \in N_y$, so $\{z_{i-1}, y\}$ is also an edge. This gives us the desired contradiction, since it implies that we have the following Hamiltonian cycle in G':

$$x,\, z_2,\, \ldots,\, z_{i-1},\, y,\, z_{n-1},\, \ldots,\, z_i,\, x$$