COMP 140: Spot It! (Writeup)

Due on September 22, 2017 at 8:00 PM

Rixner

Joel Abraham

Recipe

Pick two points that you know are the same and compute the cross product. Confirm that is is [0,0,0]. Pick two points that you know are not the same and compute the cross product. Confirm that both points lie on that line using condition (4) above. Generating all of the points in a projective plane is a multi-step process. Clearly describe the recipe for computing all of the unique points in the projective plane given a prime modulus. Clearly describe the recipe for converting points in a projective geometric plane into cards for a "Spot it!" game. You will need both the points and the prime modulus. Keep in mind the functions you wrote in part 3. You can make use of those functions here (and do not need to describe how they work, as you have already done so)!

Solution

The incidence() function takes an input of a 3-tuple representing a point, a 3-tuple representing a line, and an prime m which determines the finite field in which we are working, $\mathbb{Z}_m$. The function returns a boolean value, returning true if and only if the point is incident to the line (under some specific projective geometry). This is evaluated by determining whether $l_a * p_x + l_b * p_y + l_c * p_z \equiv 0 \mod m$. The function returns the truth value of the previous equation.

The equivalent() function is used to test if two points (lines) are equivalent. The function takes as an input two 3-tuples representing two points and a prime integer representing the *mod*, and returns a boolean value, asserting whether or not the two points are equivalent. The mathematical definition of equivalence is that points p and q are equivalent iff $(p_x, p_y, p_z) \equiv (kq_x, kq_y, kq_z) \mod m$, where m refers to the input *mod* (we will use m and *mod* interchangeably throughout this writeup). Based on the definition of equivalence, we can derive $l_a * p_x + l_b * p_y + l_c * p_z \equiv 0 \pmod m \implies k(l_a * p_x + l_b * p_y + l_c * p_z) \equiv 0 \pmod m \implies l_a * kp_x + l_b * kp_y + l_c * kp_z) \equiv 0 \pmod m$. As such, any line l incident to point $p = (p_x, p_y, p_z)$ is also incident to point $p' = (kp_x, kp_y, kp_z)$. Since this is biconditional, it must be true that if two points are collinear then they must be equivalent. And, we can test for collinearity by evaluating the cross product of the two points/vectors; if the cross product is the zero vector, then the two points must be collinear and therefore equivalent. Since $p \times q = [p_yq_z - p_zq_y, p_zq_x - p_xq_z, p_xq_y - p_yq_x]$, the two points are equivalent if and only if $p \times q = [0, 0, 0] \pmod m$. As such, the function returns the truth value of the previous equation. This can be verified with two examples:

(a) Choose $p = (1, 2, 3)$ and $q = (2, 4, 6)$ in $\mathbb{Z}_7$. Then, we use the definition of the cross product $p \times q = [p_yq_z - p_zq_y, p_zq_x - p_xq_z, p_xq_y - p_yq_x] = [2*6-3*4, 3*2-1*6, 1*4-2*2] \pmod 7 \implies p \times q = [0, 0, 0]$. Thus, p and q are equivalent in $\mathbb{Z}_7$.

(b) Choose $p = (1, 2, 3)$ and $q = (2, 4, 5)$ in $\mathbb{Z}_7$. Then, we use the definition of the cross product $p \times q = [p_yq_z - p_zq_y, p_zq_x - p_xq_z, p_xq_y - p_yq_x] = [2 * 5 - 3 * 4, 3 * 2 - 1 * 5, 1 * 4 - 2 * 2] \pmod 7 \implies p \times q = [5, 1, 0] \neq [0, 0, 0]$. Thus, p and q are distinct in $\mathbb{Z}_7$.

The function generate_all_points() takes a prime *mod* as an input and returns a list of 3-tuples, which represents the set of all unique points modulo (mod). First, the function considers all possible points modulo (mod), which is done via nested loops across all integers $p = (p_x, p_y, p_z)$ in the interval $[0, mod)$. Then, the function checks if p is the point $(0, 0, 0)$, in which case it continues iterating through points since this is an undefined point. Now that we know that $p \neq (0, 0, 0)$, we want to check if p is unique; if and only if p is unique, then we will add it to the list of points. So, we assume that p is unique and then proceed to verify this assumption. To do so, we iterate through all points q already in the list of points. p is nonunique if and only if it is nonequivalent with every point q in the list. Thus, we check for all points q in the list of points if equivalent(p, q, *mod*) is True. If it is True, then we know that p is not unique and we continue checking points. If it is False for all q in the list, then we know that p must be a unique point, so we add it to the list. Therefore, since we are considering every possible point in $\mathbb{Z}_m$, we know that all unique points

are included, and since we only add unique elements at every step of the iterative process, we know that no two elements in the output list are equivalent. This method successfully generates all unique points in the projective plane given a prime modulus.

The function create_cards() takes as an input a list of all points in the projective plane, a list of all lines in the projective plane (which is isomorphic to the list of points) and a prime *mod* indicating that we are operating in the finite field $\mathbb{Z}_{mod}$. The output of the function is a list of cards, where each card is a list of integers representing the symbols they contain (given by the index of the lines incident to a certain point in the list of lines). We process points sequentially, so we start out with an empty list of cards. Since points represent cards in Spot It! and lines represent symbols, we can find the symbols on each card by considering the lines that pass through each point. So, we first iterate through all points in the list of points. For each point, we start out with some empty card. We find the lines that pass through the point by iterating through all lines in the list of all possible lines. If a certain line is incident to the point, we add the symbol of the line (given by the index of the line) to the card. Once we've iterated through all the lines, the card should be filled with $mod + 1$ symbols. We then add this object to the list of cards and continue with the next point. Once we've determined the lines that pass through each point, we've filled all cards and we just return the completed list of cards. Now that we've completed our deck, we can now play Spot It!

Discussion

If you use a prime modulus of 5, you will generate a valid deck of 31 "Spot it!" cards, each with 6 images on them. If you use a prime modules of 7, you will generate a valid deck of 57 "Spot it!" cards, each with 8 images on them.
Suppose you wanted to create a valid deck of 40 "Spot it!" cards. Is this possible? If not, why not? If so, how would you go about doing so?

Solution
One solution would be to consider a deck of 40 cards in which all cards have 2 pictures – one universal picture U and one additional picture P_i, where $P_i \neq P_j$ if $i \neq j$. Then, the deck is given by $D = [(U, P_1), (U, P_2), ..., (U, P_{40})]$. Here, any two distinct cards have the image U in common, their additional pictures are definitionally different, and each card has exactly 2 images, thus this deck forms a valid spot it deck.
An alternative, yet undeniably less creative, solution would be to consider the deck generated using a prime modulus of 7 and removing 17 cards, yielding a valid deck with 40 cards.