Card Pile Application Report

a) When picking up the cards by rows, whichever corner we decided to start from becomes the new left top corner, and the original left top corner becomes the corner we picked the cards up from initially. Since this is the case, when we lay the cards back out and pick them up using the same specification, the left top corner returns to its original state, along with the other corners and card orderings. Thus, we only ever make two iterations of this process. In short, the maximum count values for any sized deck transformed by rows are two, as they always return to their original ordering when we apply the same transformation to the transformed deck.

However, there are some exceptions to this. When the specification is the top left, this corner never moves, and we pick up the cards in the original ordering. Hence, it is never changing, and the count is at max one. When the row length is equal to one or the size of the deck, the values of count change as the left top would equal the right top, or the left bottom would equal the right bottom, respectively. Thus, only those iterating backward in each case would require two iterations to return to the original order in such conditions.

b) The Maximum count value observed overall for the pile sizes between 1 and 20 is 18. The Pile sizes that produce this are:

-20 at specification TL: Row Length 2 and 10
-20 at specification BR: Row Length 2, 4, 5 and 10
-18 at specification BL: Row Length 6 and 9
-18 at specification TR: Row Length 2 and 3

We observed patterns and relationships between the counts of factors of the card piles, yet we could not determine a specific way to work each out, as the relationships were different each time. Some patterns we did notice were that prime numbers always give the same counts, and so do perfect squares when evaluated at the square. We have attached an additional PDF called Dataset, with the counts observed for every number 20 or less.

With the three inputs of pile size, row length, and specification, there is more than likely a mathematical or algorithmic approach to calculate the count. One path we looked into involved the nature of the pile size, where it will always be a rectangle or square. The value of count is determined by the number of iterations the algorithm must go through before the resulting function is the same as its original. As such, we can picture the count as a circle, with each function call being a point on the circle. Any specification is a variation of the circle, which implies the formula to calculate count without going through each transformation exists within the circle.

c) To determine how many accessible card piles there are in a card pile of size six, we wrote a method that performs random transformations up to a specified number to find the accessible piles. Each transformation performed on the card pile would result in a different ordering of the cards. However, a lot of these orderings would repeat.

Using an array list, we could compare the new accessible pile with previous ones, if we had not previously seen the ordering, we added it to the list. We ran the random transformations up to ten times the possible accessible piles to get a good range of data and found 48 accessible piles. To increase the chances of finding more accessible piles, we increased this number by a factor of ten multiple times, to the point where the method would take a long time to execute, and still found 48 possible accessible piles. Seeing as we were no longer finding new accessible piles, we concluded that for a card pile of size six, there are only 48 different orderings. Row lengths of three and two (the factors of six) produced the same orderings when compared.

We used the same method for card piles of sizes seven, eight, and nine. Since seven is a prime number, the only possible row lengths were one and seven. Thus, we predicted only two possible orderings of the card pile. Our method confirmed this. For a pile of size eight, the possible orderings were eight factorial, which is 40320. Out of these possible orderings, we only found 24 accessible piles. For a pile of size nine, since it has a square layout in rows of three, we theorized position five, which is exactly in the middle, would never move regardless of how we picked the cards up. We confirmed this in our test and found only eight accessible piles.

We believe it is possible to compute the accessible piles for any pile of size n. However, it may not be feasible. We can always determine the accessible piles for any n that is a prime number, which will always have only two possible orderings for reasons we have discussed previously in this report. Pile sizes that have exactly three factors always have exactly eight accessible piles. For piles larger than 14 and not these particular cases, we need a large amount of computation time as the random transformations needed to get an accurate number of accessible piles to become very large. We started by analysing ten and were able to access 1920 permutations of the pile, we moved on to 12 and found 7680. For numbers larger than 12, we find the number of accessible piles increases dramatically, and on average, the larger the test range and pile size, the longer it takes to compute accessible piles. Thus, we conclude that it is possible to compute all accessible piles for any number. However, it is only feasible to compute accessible piles within a realistic timeframe for sizes that are prime numbers, have precisely three factors, or are less than 14.

We have left the method that checks for accessible piles in the CP class file. The user can access this through stdin by loading a deck and passing Z with a row length. Example: l 6, Z 3.

Tane Carney, Kane Colvin, Samuel Ng