**COSC241 Assignment - Deal with It Report**

This report answers the questions outlined in the ‘Deal with It’ assignment handout regarding possible experiments and investigations made through the code. The questions and answers are as followed:

**(a) Consider the count values resulting from the “pick up by rows” specifications (those beginning with an L or R). What values do they take and why?**

*For this question we refer to each row as “sets”. This gives the best visual description to how the cards are picked up and transformed.*

Given the specifications of starting with either L or R, any of these will always take the values of either **1 or 2**. This is because it will always pick up the cards in a “set” of row amount *n*. For example, if *we have size = 6, rowLength = 3, spec = RB*; we will always be picking up (transforming) in “sets” of 3, meaning from 6-4 and then 3-1. This happens only once again when we place them back down as 1-3 then 4-6. Therefore, it simply takes a total of *2* *transformations* to get back to the original 1 to 6 (1 to *n*) placement of cards. In the best-case scenario the order won’t change, for example if the *size = 6, rowLength = 1, spec = RT*, the order won’t be reversed at all, as it’s literally a “set” of 1. If you reverse the “set”, it won’t change the order as it is only one number.

Overall, if it is from the left the contents of the “set” will be held in the chronological order while transforming. If it’s from the right, it’ll be in reverse. Just the same as if it’s from the top, the order will be chronological within the “set”, and if specifying from the bottom, it’ll be the reverse. The contents of the “set” will not change when starting from L or R because it isn’t ever fully “shuffled” (like it would be by going from top to bottom).

**(b) What is the maximum count value produced for any specification and any pile size of 20 or less? What pile size(s), row length(s) and specification(s) produce it? Given a pile size, row length, and specification can you think of a way of computing its count that doesn’t rely on actually carrying out that many transformations?**

The maximum count value produced for any specification and pile of 20 or less is **18**. The variable specifications that produce a maximum count of 18 are:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CP 20 2** | **CP 20 4** | **CP 20 5** | **CP 20 10** | **CP 18 2** | **CP 18 3** | **CP 18 6** | **CP 18 9** |
| TL 18 | BR 18 | BR 18 | TL 18 | TR 18 | TR 18 | BL 18 | BL 18 |
| BR 18 |  |  | BR 18 |  |  |  |  |

A potential way to compute this count without carrying out 18 transformations could be to write in an additional method that could be called. In this method a loop takes in the pile size and row length, and checks a factor of the pile size, not including 1 and the pile size itself. It’d then take the highest number of counts from that. This way you are only using multiplication and the inputted numbers to figure out the count, opposed to actually running through the transform method 18 (or *n* amount) times.

**(c) There are 720 possible card piles consisting of the numbers 1 through 6 in some order. Call such a pile *accessible* if it can be reached from the original pile 123456 by some sequence of transformations. How many *accessible* piles are there? What about seven, eight or nine card piles? For how large a value of *n* do you think it might be feasible to compute the number of accessible piles (and why)?**

There are ….. accessible piles that are there