Code Jam 2020 - Round 1B

Analysis: Join the Ranks

Test Set 1

Test set 1 contains the 12 possible inputs allowed by the limits. We can try to solve the problem via a breadth-first search (BFS), on the graph of states, where a state is the current arrangement of cards in the deck. Notice, however, that cases with 14 total cards would yield an enormous graph and make the solution run too slowly — probably even too slowly for us to run the code locally to precompute answers that we can then hardcode!

However, one observation will help us: since the success condition does not involve the suits of the cards at all, we can ignore them and work only with the ranks. That dramatically cuts down the number of possible states, by going from $(\mathbf{R} \times \mathbf{S})!$ to $(\mathbf{R} \times \mathbf{S})! / ((\mathbf{S}!)^{\mathbf{R}})$. This allows a BFS to finish fast enough for this test set. We can also use this observation while solving the next test set...

Test Set 2

For test set 2, the worst case (**R**=40, **S**=40) has around 1.8×10^{2517} unique orderings. This means that the brute force solution will not work.

Our first important observation is that the reordering operation can decrease the number of adjacent cards of different ranks by at most two. In the starting configuration there are (**R** × **S**)-1 adjacent cards of different ranks. In the ending configuration there are **R**-1 adjacent cards of different ranks. So to get from (**R** × **S**)-1 to **R**-1 we need at least ceil((**R** × **S** - **R**) / 2) operations.

Now that we know ceil((**R** × **S** - **R**) / 2) is a lower bound on the answer, if we can come up with a method that is guaranteed to use no more than ceil((**R** × **S** - **R**) / 2) steps, then it will always produce a valid answer.

Now we will outline a way to sort the cards using exactly that many operations. The invariant we maintain is that at all times, for the ranks X and Y of any two consecutive cards, either Y = X, or Y = (X + 1) mod **R**. This is of course true for the initial ordering of the deck.

We repeatedly perform the following operation, as long as the number of adjacent pairs of cards of the same rank is less than **R** - 1 and the operation would not pick up the bottom card of the deck: find the largest block of consecutive cards from the top that contains exactly 2 different ranks to use as pile A. By the invariant, this will be one or more cards with rank X, followed by one or more cards with rank (X+1) mod **R**. Then, starting from the first card from the top that is not on pile A, take as pile B the largest block of consecutive cards that does not contain any cards of rank X, plus all consecutive cards of rank X that immediately follow that block. Notice that at least one such card of rank X must exist; otherwise, by the invariant, the number of adjacent pairs of cards of different ranks would already be **R**-1.

We can show that this operation reduces the number of adjacent cards of different ranks by 2 every time it does not pick up the bottom card of the deck. To show this, notice that the bottom of pile B is a card of rank X and the first card left over in the deck is, by the invariant, (X + 1) mod **R**. That means that the new adjacent pairs are two cards of rank X (the bottom of pile B and the top of pile A) and two cards of rank (X + 1) mod **R** (the bottom of pile A and the top of the leftover deck). The broken adjacent pairs are — by definition of piles A and B — both of cards of different rank. Therefore, the number of adjacent pairs of cards of different rank decreases by 2 with this operation.

Suppose that performing the operation would pick up the bottom card of the deck. That means that all cards of rank X are in two contiguous blocks at the top and bottom of the deck before the operation is performed. In addition, since this is the first time the bottom card of the deck is picked up for an operation, X = **R**. Because of the invariant, that requires every other rank to be in a single contiguous block. In this ordering, there are exactly **R** adjacent pairs of cards with different ranks. Instead of the operation above, we finish by making pile A consist of the largest block of consecutive cards of rank **R**, starting at the top, and pile B be the rest of the deck. After performing the operation, **R** - 1 pairs of adjacent cards of different ranks remain (by a similar argument as before, ignoring the broken and created pairs that involved the leftover deck, since there is none leftover) and the final card of the deck is still **R**.

After performing the repeated operation floor((**R** × **S** - **R**) / 2) times, the number of adjacent pairs of cards of the same rank decreases to **R** - 1 if **R** × **S** - **R** is even or **R** if it's odd. Notice that the number remains greater than **R** before each such operation, so we would never have picked up the bottom card of the deck. In the even case, the number of adjacent pairs of cards of the same rank is now minimum and we never picked up the bottom card of the deck, so we are at exactly the target ordering. In the odd case, we arrive at the case in which we do pick up the bottom card of the deck with our last operation, but as argued above, that operation also leaves the deck in the target order.