Code Golfing 005 - Colorized Rummy

Maximilian Noppel

March 25, 2021

The Problem

We need to solve a subproblem of the well known card game Rummy. In Rummy the player must create groups of at least three cards. The available cards are the numbers 1 to 13 in the colors yellow (r), blue (b), black (k) and red (r). The player is allowed to either make a group from cards of the same colors but in ascending order or of different colors but with the same numbers. In the second option the biggest possible group is therefore of size four, as only four colors exist. In this Code Golfing we only consider the second case. We assume every card has the same number. The problem is a decision problem. Given a string of the letters `y`,`b`,`k` and `r`, each representing a card of the given color, the program must decide if the cards can be played validly. If this is the case it should output 0 for `OK`. Otherwise the output should be 1 for `ERROR`.

Intuition

Give is string $s \in \{\texttt{y},\texttt{b},\texttt{k},\texttt{r}\}^*$. It turns out that the order does not matter. To solve this task, we only need the number of occurrences of each letter. The counting can be done in $O(n)$. So the input to our program is $\mathbb{R}^4$.

In the first step we sort our four numbers. W.l.o.g. this could look like in Fig. 1.

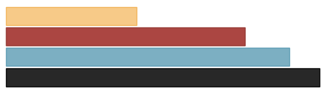

Figure 1. Sorted number of cards

The idea of our solution is that we can put the color with the least occurrences into the gaps of the other colors. This would look like in Fig. 2.

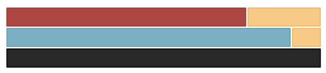

Figure 2. Put `y` into the gaps

This is a valid solution as we show in Fig. 3.

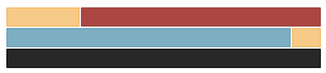

Figure 3. This yield a valid solution

In this special case the solution is clear. But what happens if we have a very small difference between the two colors that have the least occurrences, as visualized in Fig. 4.

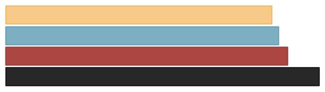

Figure 4. Small difference between `y` and `b`

In this case we have a number of groups with four different cards. It is clear that yellow could not exceed the red bar. If this would be the case, yellow would no longer have the least occurrences and the sorting would be different. Or, if all color occur equally often, the solution is also 0 `OK`.

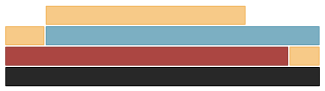

Figure 5. Sorted number of cards

The final question is how we solve the decision problem? The solution is very simple after the presented insights. W.l.o.g. yellow must be longer or equally long as the two gaps. Otherwise there would be groups with only two cards. As we must output `0` for the `OK` case, we must invert this inequality. If yellow is shorter as the two gaps combined, the problem is not solvable and the expected output is `1 ERROR`.

Some Math Given $c \in \mathbb{R}^4$ (a 4D vector). The elements of c are named $c_0, ..., c_3$ s.t. $c_0 \leq c_1 \leq c_2 \leq c_3$. Our solution is constant:

$$c_0 < 2 \cdot c_3 - c_2 - c_1$$

Solutions

In Python the solution looks as follows:

```python
import sys
s = sys.argv[1]
c = [s.count(x) for x in "yrbk"]
c=sorted(c)
exit(c[0]<2*c[3]-c[1]-c[2])
```

The winning Ruby solution is 69 bytes long:

```ruby
s=$*[0].chars.tally.values+[0,0,0].sort
exit(s[3]>=2*s[0]-s[1]-s[2])
```

The complexity is therefore only $O(n)$, where n is the number of cards. As the number of distinctive colors is constant, the complexity of sorting is also constant. This leads to the observation that if we would get directly the numbers as inputs our complexity would be $O(1)$. A rather surprising solution to this problem.

Conclusion

At the first look, the problem seems complicated. If we take some time and try to represent things visually we naturally find out that sorting of the occurrences is a good idea. From this we quickly arrive at this geometrical solution. Naive approaches that do not consider sorting end up using a lot of `if`s. Other solutions try to group things together, one group after another, by decrementing the numbers piece by piece. These solutions are highly inefficient and contain a lot of loops.

In this short article we showed how to arrive at the *optimal* solution to this problem.