project1.pdf

Dario Maddaloni (233187)

March 2022

1 invert

The function invert performs the inverse of the shuffle described in the .pdf assigned. In particular it behaves as follows:

$$invert(m,k) = \begin{cases} m & k \geq 2k \\ \frac{m}{2} & m \equiv 1 \text{ mod} 2 \wedge \; k < 2k \\ \frac{m}{2} + k & m \equiv 0 \text{ mod} 2 \wedge \; k < 2k \end{cases}$$

where k is the round of the shuffling we want to invert and m is the position of the card we are interested in.

2 complete_shuffle

First of all, we invert the position of the first three cards.
By the fact that we are considering exactly the card at the first three positions, we assign

$$s = k \quad \text{and} \quad f = 1$$

Then we have to compute iteratively the inverse of this positions until the deck will be reorganized. But, by the fact that we are interested only in three cards, we do not compute the position of all the other cards in the deck.
We will update the value of s and f when the position of the element a (which was the one at the top of the deck at the end of the shuffling) is the top one.
Now, we can return the elements: we have to return $a+1$, $b+1$, $c+1$ by the fact that we were considering the position and not the value (that starts from 1 instead of 0); $s-1$ because we are interested in the position of the element before the shuffling.

3 timing

By the fact that we always have to perform an entire cycle "for" of range k-1 (that we can approximate to k reasonably). By the fact that we can assume that

the function "invert" is an O(1) (i.e. it has a constant cost): the "if statement" has a constant time cost. We can deduce that complete-shuffle function is an O(k), where k is the number of shuffle we are interested in. Moreover, it is an O(n) by the fact that O(n)=O(2k)=O(k). So it increase linearly in dependence of the number of shuffles.
Indeed, if we apply the following python code:

```
from project1 import complete_shuffle
from time import time
for k in range(10**4,10**5,10**4):
        tic = time()
        complete_shuffle(k)
        elapsed = time() - tic
        print(f "With {k} shuffles: {elapsed:6.4e} seconds.")
```

Output:

```
With 10000 shuffles: 1.9620e-02 seconds.
With 20000 shuffles: 2.7325e-02 seconds.
With 30000 shuffles: 1.5199e-02 seconds.
With 40000 shuffles: 4.8162e-02 seconds.
With 50000 shuffles: 4.9964e-02 seconds.
With 60000 shuffles: 4.9944e-02 seconds.
With 70000 shuffles: 6.7597e-02 seconds.
With 80000 shuffles: 6.4417e-02 seconds.
With 90000 shuffles: 8.2826e-02 seconds.
```