# 1 | Fifteen Square Puzzle

From the definition of the problem, we have the state $(A, B)$:

- $A$ is a list $(a_1 \dots a_{15})$ with all the values skipping the empty square
- $B$ is a tuple $(X, Y)$ containing the coordinates of the empty square

We also define "out of order" as pairs of not-necessarily-continuous values that are not strictly increasing, and "parity" as $mod\ 2$ of the number of out-of-order pairs plus the row number of the empty square.

## 1.1 | Defining Transitions

For every single case, there is four possible transitions to make

1. Move empty square up
2. Move empty square down
3. Move empty square left
4. Move empty square right

## 1.2 | Proving Invariant

We will show that the base state has a specific parity. At $((1 \dots 15), (4, 4))$, the starting base state, it has parity $0 + 4 = 0(mod\ 2)$.

Let's declare parity $= 0\ (mod\ 2)$ as the invariant.

## 1.3 | Proving Invariant through Transitions

Let's prove that invariant is invariant through all transitions. We will do so in pairs, as the "moving square" operation is isomorphic by up-down and left-right pairs.

### 1.3.1 | Moving Up-Down

Moving the empty square up-down constitutes adding/removing three pairs of out-of-order items—shifting a empty square up would result in the item three-items-back to be moved ahead by three items.

Adding three out-of-order pairs, plus subtracting one row from the empty square position, would result in a change in parity of $3-1 = 0\ (mod\ 2)$. It follows that reversing the operation would result in $-3+1 = 0\ (mod\ 2)$.

Shifting up/down does not change the invariant.

### 1.3.2 | Moving Left-Right

Moving an empty square left-right neither changes the row number for the empty row nor the order of the items. Hence, it does not change the items that constitute the parity—making the parity the same and invariant.

### 1.4 | **Proving Invariant to End**

At the final end state, there is $\binom{15}{2}$ out of order pairs (choose any of the tiles, choose any other one, swapping the order would be the same choice, but choosing two of the same is not.)

Hence:

$$\frac{15!}{2!(13!)} = \frac{15 \times 14}{2} = 105 \tag{1}$$

At the final state, the empty row is at row $4$. $4 + 105 = 1 \ (mod\ 2)$, loosing the invariant.

By Floyd's invariant method, we see that the final state is not in invariant and hence **not** reachable from the base state via the Moving Up-Down and Moving Left-Right operations.

## 2 | **Fast Exponentiation**

### 2.1 | **Analyzing Traditional Exponentiation**

The running time of traditional exponentiation is $O(k)$, where $k$ is the power by which to raise. This is simply because an increase in the power of $k$ will result in a linear $+1$ increase in the number of multiplications.

### 2.2 | **Fast Exponentiation as FSM**

Base state: $(a, 1, b)$.

At every state $(x, y, z)$:

If $z$ is odd:

- Set $y = xy$
- Set $z = \frac{z-1}{2}$

If $z$ is even:

- Set $z = \frac{z}{2}$

Finally, $x = x^2$

Furthermore, $z$ is our derived variable counting down to $0$. If $z = 0$, $y$ is the result returned.

### 2.3 | **FSM Proof**

We will set the fact that $x^z y = a^b$ as our invariant. At the base state, where $y = 1$, $x = a$ and $z = b$, this is trivially true.

At every state update, the value of $y$ either stays unchanged ($z$ is even) or is scaled by $x$ ($z$ is odd).

In the former case, our new value of $x = {x_0}^2$ and $z$ is divided by half. Therefore, the new state update would be:

$$x^z y = {x_0}^{2\frac{z}{2}} y = {x_0}^z y = a^b \tag{2}$$

So we can see, then that we still maintain the invariant.

In the latter case, our new value of $x = {x_0}^2$, $y$ is scaled by $x_0$, and $z$ is floored and divided by half.

In the latter case, the new value of $x$ — $x^2$, per every step — is also $0 \ (mod \ y)$ as we can separate the new value of y into $y = xy_0$, which the former $x$ term dividing out one of the two $x$ in $x^2$ and the $x$ remaining known to be $0 \ (mod \ y_0)$.

Hence, the condition $x \ (mod \ y)$ is shown for all state updates and bases state to be invariant, completing the proof by Floyd's invariant method.

## 2.4 | **The Algorithm Terminates**

If we set $z$ as our derived variable, and $\forall z > 1$, we see that every step converges $z$ to the integer divisor of itself to two. At $z = 1$, $z$ is set to $0$. Therefore, the state machine follows a strictly descending derived variable which converges, meaning the algorithm terminates.

The new running time of the exponentiation, as we are constantly dividing by $2$ on the derived variable count, is $O(log(k))$ — where $k$ is the power by which to raise, a significant increase to the $O(k)$ implementation.