Доказательство, что *RLL* 2,7 эффективнее по количеству смен знака на бит информации, чем *MFM*

1. Определим задачу

Пусть существуют дискретные случайные величины A и B, принимающие значения $\{\alpha_i | i \in (0, n)\}$ и $\{\beta_i | i \in (0, m)\}$.
Конкретные значения A и B для удобства будем называть символами.
Данные для чтения или записи представляются в виде последовательности символов $\vec{\alpha}$ = $(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n)$.
Данные на ПУ (например, на жестком диске) представляются в виде последовательности символов $\vec{\beta}$ = $(\beta_0, \beta_1, \beta_2, \beta_3, \ldots, \beta_n)$.
Тогда запись на ПУ осуществляется через отображение $F : A \to B$.

Например, есть последовательность на запись 00000001000100100010. Разбиваем её на символы: 000, 000, 010, 0010, 010, 0010. Далее по отображению, определённому в RLL, записываем на диск. Например, символ 000 будет записан как NNNTNN, где T означает "смена знака есть", а N - "смены знака нет".

Определена функция $W(x)$, значение которой есть количество изменений знака на бит в символе.

Например, $W(000) = 1/3$.

Задача: доказать, что *RLL* 2,7 имеет $M[W(A)]$ меньше, чем у *MFM*, где $M[W(A)]$ есть математическое ожидание функции $W(x)$ случайной величины A.

2. Найдём значения $W(x)$ для $\forall \alpha$ и вероятности появления α_i для $\forall i$ для *RLL* и *MFM*.

Отображение *RLL* 2,7 $F : A \to B$:
$11 \to TNNN$
$10 \to NTNN$
$000 \to NNNTNN$
$010 \to TNNTNN$
$011 \to NNTNNN$
$0010 \to NNTNNTNN$
$0011 \to NNNNTNNN$

Вероятность появления символа α есть произведение вероятностей его битов. Например, для символа 11 вероятность равна вероятности единицы умножить на вероятность единицы.

```python
import pandas as pd

data = {
    "aplha_i": ["11", "10", "011", "010", "000", "0010", "0011"],
    "p_i": [1 / 4, 1 / 4, 1 / 8, 1 / 8, 1 / 8, 1 / 16, 1 / 16],
    "W(alpha_i)": [1 / 2, 1 / 2, 1 / 3, 2 / 3, 1 / 3, 1 / 2, 1 / 4]
}

rll_stats = pd.DataFrame(data)
```

rll_stats есть данные об отображении F у *RLL* 2,7

```python
rll_stats
```

	aplha_i	p_i	W(alpha_i)
0	11	0.2500	0.500000
1	10	0.2500	0.500000
2	011	0.1250	0.333333
3	010	0.1250	0.666667
4	000	0.1250	0.333333
5	0010	0.0625	0.500000
6	0011	0.0625	0.250000

Убедимся, что вероятности в сумме дают единицу

```python
rll_stats['p_i'].sum()
```

```
1.0
```

Отображение MFM $F : A \to B$:
$1 \to TN$
$(1)0 \to NN$
$(0)0 \to TN$

```python
data = {
    "aplha_i": ["1", "(0)0", "(1)0"],
    "p_i": [1 / 2, 1 / 4, 1 / 4],
    "W(alpha_i)": [1, 1, 0]
}

mfm_stats = pd.DataFrame(data)
```

mfm_stats есть аналог *rll_stats* для *MFM*

```python
mfm_stats
```

	aplha_i	p_i	W(alpha_i)
0	1	0.50	1
1	(0)0	0.25	1
2	(1)0	0.25	0

3. Найдём $M[W(A)]$ для *RLL* 2, 7 и *MFM* и сравним

Математическое ожидание функции $W(x)$ дискретной случайной величины A находится по формуле $M[W(A)] = \sum_{i=0}^{n} W(\alpha_i) * p_i$

```python
rll_ev = (rll_stats["p_i"] * rll_stats['W(alpha_i)']).sum()
mfm_ev = (mfm_stats["p_i"] * mfm_stats['W(alpha_i)']).sum()
```

Мат. ожидание *RLL* 2,7 =

```python
rll_ev
```

```
0.4635416666666667
```

Мат. ожидание *MFM* =

```python
mfm_ev
```

```
0.75
```

Отношение *MFM* к *RLL* =

```python
mfm_ev / rll_ev
```

```
1.6179775280898876
```

Как видим, МО *MFM* больше МО *RLL* примерно в полтора раза. Что и требовалось доказать.