As discussed, the runs of b and w pixels are coded using Golomb coding. For example, we would code wwwbbbbbbwbbbb as w:3,5,1,4. However, since all the numbers will be >=1, we can instead subtract 1 from all lengths and code w:2,4,0,3. This aligns better with Golomb codes since Golomb codes code integers that are >= 0.

Then, the distribution of the integers to be coded is as follows. The probability that the integer to be coded is equal to *i* is $p_{ww}^{i}(1-p_{ww})$ if we are talking about lengths of white runs, and the formula for black runs is analogous (the formula in the notes and slides has i-1 instead of i because we subtracted 1). In the notes, we say that Golomb coding, with $M = \left\lceil \frac{1}{1-p_{ww}} \right\rceil$, is "optimal".

Here is a hint of why Golomb coding is good, using the example $p_{ww} = 7/8$ that we discussed in class. The table below has several columns, which stand for: the integer to be coded (which is the length of the white run minus 1), its probability according to the formula above, what Shannon's theorem says is the optimal number of bits to code the integer, the Golomb code with $M = \left\lceil \frac{1}{1-p_{ww}} \right\rceil = \frac{1}{1-\frac{7}{8}} = 8$, and the length of the Golomb code. Clearly, the length of the Golomb code is typically very close to what Shannon's formula requires.

value	prob.	Shannon	Golomb	Length
0	0.125	3.00	0000	4
1	0.109	3.19	0001	4
2	0.096	3.39	0010	4
3	0.084	3.58	0011	4
4	0.073	3.77	0100	4
5	0.064	3.96	0101	4
6	0.056	4.16	0110	4
7	0.049	4.35	0111	4
8	0.043	4.54	10000	5
9	0.038	4.73	10001	5
10	0.033	4.93	10010	5
11	0.029	5.12	10011	5
12	0.025	5.31	10100	5
13	0.022	5.50	10101	5
14	0.019	5.70	10110	5
15	0.017	5.89	10111	5
16	0.015	6.08	110000	6
17	0.013	6.27	110001	6
18	0.011	6.47	110010	6
19	0.010	6.66	110011	6
20	0.009	6.85	110100	6