Given a vector of weights (w_i), Gumbel max sampling is a way to sample from a categorical distribution with the specified weights. Given a sequence (u_i) of iid uniform samples, let

$$g_i = -\log(-\log u_i) \ .$$

Then for each i, $g_i \sim \text{Gumbel}()$. Sampling then reduces to computing

$$x = \arg\max(\log w_i + g_i) \ .$$

The arg max computation is done in terms of comparisons. Suppose

$$\log w_1 + g_1 < \log w_2 + g_2 \ .$$

We can rewrite this to be more efficient:

$$\begin{aligned}
\log w_1 + g_1 &< \log w_2 + g_2 \\
\log w_1 - \log(-\log u_1) &< \log w_2 - \log(-\log u_2) \\
\log\left(-\frac{w_1}{\log u_1}\right) &< \log\left(-\frac{w_2}{\log u_2}\right) \\
-\frac{w_1}{\log u_1} &< -\frac{w_2}{\log u_2} && \text{(log is increasing)} \\
\frac{w_2}{\log u_2} &< \frac{w_1}{\log u_1} && \text{(negation is decreasing)} \\
w_2 \log u_1 &< w_1 \log u_2 && \text{(log u_1 * log u_2 is positive)}
\end{aligned}$$

In Rust, we can write this as

```
pub fn pflip(weights: &[f64], rng: &mut impl Rng) -> usize {
    assert!(!weights.is_empty(), "Empty container");
    weights
        .iter()
        .map(|w| {
            (w , rng.gen::<f64>().ln())
        })
        .enumerate()
        .max_by(|(_, (w1, l1)), (_, (w2, l2))|
        (*w2 * l1).partial_cmp(&(*w1 * l2)).unwrap())
        .unwrap()
        .0
}
```