The coin problem (also known as bias amplification) involves distinguishing, with probability at least 2/3, between n i.i.d. coins which are heads with probability $(1/2+\beta)$ from ones which are heads with probability $(1/2-\beta)$. We are interested in the space complexity of the coin problem, corresponding to the width of a read-once branching program solving the problem.

The coin problem becomes more difficult as $\beta$ becomes smaller. Statistically, it can be solved whenever $\beta = \Omega(n^{-1/2})$, using counting. It has been previously shown that for $\beta = O(n^{-1/2})$, counting is essentially optimal (equivalently, width $\text{poly}(n)$ is necessary [Braverman-Garg-Woodruff FOCS'20]). On the other hand, the coin problem only requires $O(\log n)$ width for $\beta > n^{-c}$ for any constant $c > \log_2(\sqrt{5}-1) \approx 0.306$ (following low-width simulation of AND-OR tree of [Valiant JAlg'84]).

In this work, we close the gap between the bounds, showing a tight threshold between the values of $\beta = n^{-c}$ where $O(\log n)$ width suffices and the regime where $\text{poly}(n)$ width is needed, with a transition at $c = 1/3$.

This gives a complete characterization (up to constant factors) of the memory complexity of solving the coin problem, for all values of bias $\beta$.

We introduce new techniques in both bounds. For the upper bound, we give a construction based on recursive majority that does not require a memory stack of size $\log n$ bits. For the lower bound, we introduce new combinatorial techniques for analyzing progression of the success probabilities in read-once branching programs.

The talk is based on joint work with Mark Braverman and Sumhega Garg.