I made an interesting, and seemingly logical, observation on BAMs with a small number of memorized patterns. The number of filled bits competing with each other seems to matter when recalling a pattern. In other words, if you try to recall 2 A patterns at the same time by submitting both of the B patters simultaneously, then the number of 1s in the B pattern for each relation will determine which A pattern is recalled. If 1 pattern is [1,0,1,0,0,0], [1,0,0,0] and the other is [0,0,0,0,1,1], [0,1,1,1], and you submit [1,1,1,0] then the BAM will recall [0,0,0,0,1,1], [0,1,1,1]. On the surface, as this is about as deep as one can go with my understanding, this appears to be because a pattern with more 1s will create “deeper” holes in the matrix for the BAM to fall into. Although, thinking about it more now, it might also be that it creates a broader hole that increases the likelihood of a pattern matching.

I also found that when adding a medium number of patterns to the BAM, despite being within the technical limits of the BAM, the accuracy of the BAM decreased drastically. Despite giving perfect input, the BAM would no longer return the proper memorized pair. This could be a problem with my implementation, or it could just be a limitation of the BAM and the patterns the I memorized.