

CLAIMS

WHAT IS CLAIMED IS:

1. A system, comprising:
 - one or more processors; and
 - memory to store executable instructions that, if executed by the one or more processors, cause the system to:
 - receive a request to remove an entry from a Bloom filter;
 - query the Bloom filter to determine a current iteration value that is either from a first set of iteration values or from a second set of iteration values, the first set of iteration values indicating presence of the entry in the Bloom filter and the second set of iteration values indicating the entry having been removed from the Bloom filter; and
 - respond to the second request by at least:
 - if the current iteration value corresponds to the second set of iteration values, indicating that the entry is not present within the Bloom filter; and
 - if the current iteration value corresponds to the first set of iteration values:
 - incrementing the current iteration value to a new iteration value corresponding to the second set of iteration values;
 - utilizing the new iteration value and the entry to generate an output value; and
 - adding the output value to the Bloom filter to indicate removal of the entry from the Bloom filter.
2. The system of claim 1, wherein the executable instructions further cause the system to query the Bloom filter multiple times, each with a different output value corresponding to a different iteration value to determine the current iteration value as a highest iteration value for the entry.
3. The system of claim 1, wherein the first set of iteration values comprises first iteration values that each have a property of even integers and the second set of iteration values comprises second iteration values that each have a property of odd integers.

4. The system of claim 1, wherein the executable instructions further cause the system to:

receive a second request to add the entry to the Bloom filter;

query the Bloom filter to determine the current iteration value for the entry; and

respond to the second request by at least:

if the current iteration value corresponds to the first set of iteration values, indicating that the entry is present in the Bloom filter; and

if the current iteration value corresponds to the second set of iteration values:

incrementing the current iteration value to a second new iteration value corresponding to the first set of iteration values;

utilizing the second new iteration value and the entry to generate a second output value; and

adding the second output value to the Bloom filter to indicate that the entry is present in the Bloom filter.

5. A computer-implemented method, comprising:

querying a probabilistic data structure to determine a current iteration state for an entry in the probabilistic data structure, the current iteration state indicating a state of the entry from a plurality of states that include a first state corresponding to the entry being a member of a set and a second state corresponding to absence of the entry from the set;

as a result of the current iteration state corresponding to the entry being a member of the set, determining a new iteration state based at least in part on the current iteration state;

utilizing the new iteration state and the entry to generate an output value; and

adding the output value to the probabilistic data structure to indicate, in the probabilistic data structure, removal of the entry from the set.

6. The computer-implemented method of claim 5, wherein:

the probabilistic data structure is a Bloom filter; and

adding the output value to the Bloom filter includes changing one or more bits within the Bloom filter from zero to one to denote presence of the output value within the Bloom filter.

7. The computer-implemented method of claim 5, wherein the method further comprises:

receiving a request to add the entry to the probabilistic data structure;

querying the probabilistic data structure to determine the current iteration state for the entry; and

as a result of the current iteration state corresponding to the second state:

determining a different iteration state based at least in part on the current iteration state and corresponding to the first state;

utilizing the different iteration state and the entry to generate a second output value; and

adding the second output value to the probabilistic data structure to indicate, in the probabilistic data structure, addition of the entry to the set.

8. The computer-implemented method of claim 5, wherein the current iteration state has a parity attribute that is either a first parity state or a second parity state, wherein the first parity state corresponds to the first state and the second parity state corresponds to the second state.

9. The computer-implemented method of claim 8, wherein the first parity state is a property of odd integers and the second parity state is a property of even integers.

10. The computer-implemented method of claim 5, wherein the output value is generated by inputting at least the new iteration state and the entry into a hash function.

11. The computer-implemented method of claim 5, wherein querying the probabilistic data structure to determine the current iteration value includes:

querying the probabilistic data structure multiple times, each with a different output value corresponding to a different iteration state and the entry; and

determining the current iteration state as the highest iteration state for which the entry was added to the set.

12. The computer-implemented method of claim 5, wherein removal of the entry from the set is performed in response to a request generated as a result of removal of the entry from a database associated with the probabilistic data structure.

13. A non-transitory computer-readable storage medium having stored thereon executable instructions that, if executed by one or more processors of a computer system, cause the computer system to at least:

determine a current iteration value for an entry in a probabilistic data structure, the current iteration value indicating a state of the entry from a plurality of states that include a first state and a second state, the current iteration value indicating a number of times the entry was used to update the probabilistic data structure, wherein at least one possible value of the current iteration value indicates whether the entry has been removed from a set;

determine the state of the entry based at least in part on the current iteration value; and
provide information indicating the state.

14. The non-transitory computer-readable storage medium of claim 13, wherein the instructions further cause the computer system to query the probabilistic data structure multiple times, each with a different output value corresponding to a different iteration value and the entry to determine the current iteration value.

15. The non-transitory computer-readable storage medium of claim 13, wherein the first state corresponds to membership of the entry in the set and the second state corresponds to absence of the entry in the set.

16. The non-transitory computer-readable storage medium of claim 13, wherein the probabilistic data structure is a Bloom filter comprising one or more bits used to denote presence of output values within the Bloom filter, where each output value of the output values corresponds to the entry and an iteration value.

17. The non-transitory computer-readable storage medium of claim 13, wherein the instructions further cause the computer system to:

receive a request to add the entry to the probabilistic data structure;

query the probabilistic data structure to determine the current iteration value for the entry; and

fulfill the request by at least:

if the current iteration value corresponds to the first state, indicating that the entry is a member of the set; and

if the current iteration value corresponds to the second state:

incrementing the current iteration value to be a new current iteration value corresponding to the first state;

utilizing the new current iteration value and the entry to generate an output value; and

adding the output value to the probabilistic data structure to indicate, in the probabilistic data structure, addition of the entry to the set.

18. The non-transitory computer-readable storage medium of claim 13, wherein the instructions further cause the computer system to:

receive a request to remove the entry from the probabilistic data structure, the request generated in response to detection of removal of the entry from a database associated with the probabilistic data structure;

query the probabilistic data structure to determine the current iteration value for the entry; and

fulfill the request by at least:

if the current iteration value corresponds to the second state, indicating that the entry is not present in the set; and

if the current iteration value corresponds to the first state:

incrementing the current iteration value to be a new current iteration value corresponding to the second state;

utilizing the new current iteration value and the entry to generate an output value; and

adding the output value to the probabilistic data structure to indicate, in the probabilistic data structure, removal of the entry from the set.

19. The non-transitory computer-readable storage medium of claim 13, wherein the current iteration value has a parity attribute that is either a first parity value or a second parity value, wherein the first parity value corresponds to the first state and the second parity value corresponds to the second state.

20. The non-transitory computer-readable storage medium of claim 13, wherein the instructions further cause the computer system to:

as a result of the current iteration value corresponding to the entry being present in the probabilistic data structure, select a new iteration value corresponding to the entry being absent from the probabilistic data structure;

generate, using the new iteration value and the entry as input to a function, a second output value; and

add the second output value to the probabilistic data structure.