Bloom Filter and Cuckoo Filter on N-ary Trees

Vedanshi Shah

2022 May 10

Overview

We consider the problem of searching for a particular data element x in the set. If we simply take the leaf nodes and represent them as a list of elements, and then search this list, we would obtain a Boolean answer of whether x is in the data set or not. We would not find out any information related to the tree structure of the data: where x is in the tree and any of the metadata associated with x. We would lose vital information encapsulated in the tree organization, and therefore lose its original intent and advantage. To retain the tree location and metadata for x, we must conduct the search on the tree. The naive way of doing a breadth-first or depth-first tree traversal would yield x's location(s) in the tree and hence all its associated metadata. The cost of such a traversal is that every node in the tree must be visited and every leaf node also compared with x.

At each interior node, I compute a bloom filter for the entire subtree rooted at this node. By checking the bloom filter, we can eliminate the possibility that x is in that subtree and altogether forego the traversal or search in that subtree. This way, every subtree not having x as one of its leaf nodes does not get traversed or searched at all, thus saving the unnecessary cost.

Bloom Filters

A Bloom filter is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set. False positive matches are possible, but false negatives are not – in other words, a query returns either "possibly in set" or "definitely not in set". Elements can be added to the set, but not removed (though this can be addressed with the counting Bloom filter variant); the more items added, the larger the probability of false positives.

An empty Bloom filter is a bit array of m bits, all set to 0. There must also be k different hash functions defined, each of which maps or hashes some set element to one of the m array positions, generating a uniform random distribution.
To add an element, feed it to each of the k hash functions to get k array positions. Set the bits at all these positions to 1.
To query for an element (test whether it is in the set), feed it to each of the k hash functions to get k array positions. If any of the bits at these positions is 0, the element is definitely not in the set; if it were, then all the bits would have been set to 1 when it was inserted. If all are 1, then either the element is in the set, or the bits have by chance been set to 1 during the insertion of other elements, resulting in a false positive.

Further Optimization (Using Cuckoo Filters

]A cuckoo filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set, like a Bloom filter does. False positive matches are possible, but false negatives are not – in other words, a query returns either "possibly in set" or "definitely not in set". A cuckoo filter can also delete existing items, which is not supported by Bloom filters. In addition, for applications that store many items and target moderately low false positive rates, cuckoo filters can achieve lower space overhead than space-optimized Bloom filters.

Data Structures Used

- N-ary Trees
- Hashmaps
- Stacks

References

- https://american-cse.org/csci2015/data/9795a018.pdf
- https://www.cs.cmu.edu/~dga/papers/cuckoo-conext2014.pdf
- https://www.geeksforgeeks.org/generic-treesn-array-trees/