

**Sampling and Reconstruction Using Bloom Filters**

**Abstract:**

In this paper, we address the problem of sampling from a set and reconstructing a set stored as a Bloom filter. To the best of our knowledge our work is the first to address this question. We introduce a novel hierarchical data structure called BloomSampleTree that helps us design efficient algorithms to extract an almost uniform sample from the set stored in a Bloom filter and also allows us to reconstruct the set efficiently. In the case where the hash functions used in the Bloom filter implementation are partially invertible, in the sense that it is easy to calculate the set of elements that map to a particular hash value, we propose a second, more space-efficient method called HashInvert for the reconstruction. We study the properties of these two methods both analytically as well as experimentally. We provide bounds on run times for both methods and sample quality for the BloomSampleTree based algorithm, and show through an extensive experimental evaluation that our methods are efficient and effective.

**Existing System:**

We note that other compact structures, such as sketches, have been used as compact storage structures from which samples can later be obtained. However, a limitation of this approach is that the sketches that are proposed to be created are specifically for the problem of sampling and tend to be output sensitive in their design (and do not support reconstruction). Our work, on the other hand, shows how to draw samples as well as reconstruct sets from a widely-used generic synopsis structure, the Bloom filter, that is also useful for several other applications.

**Proposed System:**

(i) We introduce a novel data-structure called BloomSample-Tree that can be used to sample from a set stored in a Bloom filter as well as reconstruct that set. This is the first method to efficiently solve this problem. Moreover, the BloomSampleTree takes into account the occupancy of the namespace and can change size as the occupancy changes,

(ii) We provide theoretical bounds on the runtime and on the quality of samples generated by our BloomSampleTree-based algorithm, show them to be near-uniform.

(iii) We show through extensive evaluations that our Bloom- SampleTree-based algorithms are efficient and provide good quality samples.