CSC 591

Advanced Algorithms **H1**

Submitted by **Jayant Dhawan (jdhawan2)**

Question 1

Explain the Bloom filter principle: "Wherever a list or set is used, and space is at a premium, consider using a Bloom filter if the effect of false positives can be mitigated."

A Bloom filter is a space-efficient data structure that is used to represent a set of values. It uses *k* hash functions to map each of *n* values from the input set into *k* hashed values in the range [1,*m*]. These hashed values are then stored in an array of *m* bits, initialized to 0, by storing a 1 at the array index corresponding to the hashed value. The Bloom filter thus represents an arbitrary size input list with a smaller sized array resulting in significant space savings.

The one drawback of Bloom filters is the issue of a false positive - an event in which an element that is not present in the list represented by the Bloom filter but is indicated to be present because all the bits at the array indices corresponding to its hashed values are 1. The rate of having false positives can be controlled by choosing hash functions for tweaking the values of the number of hash functions, *k*, and the size of the Bloom filter, *m*. So Bloom filters are extremely useful in applications where space savings would be advantageous, but these advantages must be weighed against the effects of a finite possibility of having false positives.

Question 2

Explain why deletions are not allowed on standard Bloom filters and describe a solution that has been proposed to address this limitation.

Deleting an entry from a Bloom filter would involve hashing the element to be deleted, and setting the corresponding bits at the array indices in the Bloom filter to 0. These bit positions might have corresponded to the hashed values of some other element in the set. The Bloom filter thus no longer correctly represents all elements in the input set, and this might result in false negatives. Due to this unintended effect, deletions are not allowed in standard Bloom filters.

A solution to this problem involves a *counting* Bloom filter in which each element of the Bloom filter is a counter, instead of a single bit. Inserting an item involves incrementing the counters at the corresponding array indices, and deleting an item involves decrementing those counters.

Question 3

Describe three network applications of Bloom filters. For each of these applications, explain why the use of Bloom filters is recommended.

Distributed Caching

Bloom filters are used for representing Web caches. In a system with Web proxies, when a Web page is requested by the client, and it's not present in the proxy's cache, the proxy attempts to determine if the page is present in another proxy's cache. In case it is, a request is made to that proxy instead of fetching the page from the Web. So each proxy must be aware of the list of Web pages available in other proxies' caches.

To significantly reduce network traffic, each proxy periodically broadcasts a Bloom filter representing the contents of its cache to all other proxies. Doing a lookup in other proxies' cache therefore involves checking the Bloom filters for the other proxies. One shortcoming to this approach is that when a false positive occurs, a proxy may request a Web page from another proxy which does not really have the page in its cache, resulting in an additional delay. It is not a major concern, though, because even without using Bloom filters, false positives and false negatives are a possibility since the cache may not be updated when a query is made.

Packet Routing: Detecting Loops

In multicast and unicast network protocols, forwarding loops are dealt with by using the IP's Time-To-Live (TTL) field in the packet header. The TTL is a counter that is decremented each time the packet is forwarded. Once the counter becomes 0 due to multiple hops, the packet gets dropped. In small loops, such as those encountered in experimental protocols, Bloom filters may be used instead. Each packet's header would contain a Bloom filter to keep track of the nodes visited during its hops. When the packet reaches a node, a mask present at the node is ORed with the Bloom filter contained in the packet. If the filter does not change, which means that the bit positions were already 1, that implies (barring false positives) that the node was already visited and so there may be a loop.

Multicast

In a multicast protocol, a router determines the interfaces to which a packet destined to a multicast address is to be forwarded to, by referring to the list of interfaces associated with that multicast address. The problem with this approach is that the router needs to maintain the list of interfaces corresponding to each multicast address. An approach using Bloom filters suggests that each interface has a Bloom filter of the multicast addresses associated with it. So when a packet arrives on one interface, having a multicast address, the Bloom filters for all other interfaces are checked to determine whether they are associated with that multicast address, thus eliminating the need for storing the list at the router.

References

1. Broder, A., & Mitzenmacher, M. (n.d.). Network Applications of Bloom Filters: A Survey. *Internet Mathematics,* 485-509.