CS 531: Homework 3, Fall 2024
Due October 8, 2024 at 3:00 PM

Study Group [10]

Provide the names of the members of your study group. If you do not have a study group, explicitly write ”None.”

Bloom Filters [30]

1. Bloom filters are close cousins of hash tables and are suitable for applications requiring fast lookups with a dynamically changing set of objects. Compared to hash tables, Bloom filters are extremely space-efficient but, in exchange, they occasionally make errors. The data structure of Bloom filters maintains an n-bit string, or equivalently a length-n array A in which each entry is either 0 or 1. (All entries are initialized to 0.) The structure also uses m hash functions $h_1, h_2, \ldots, h_m$, each mapping the universe $\mathcal{U}$ of all possible keys to the set $\{0, 1, 2, \ldots, n-1\}$ of array positions. The parameter m is proportional to the number of bits that the Bloom filter uses per insertion, and is typically a small constant. Every time a key is inserted into a Bloom filter, each of the m hash functions plants a flag by setting the corresponding bit of the array A to 1:

```
    for i = 1 to m do
        A[h_i(key)] := 1
```

   In the LOOKUP operation, a Bloom filter looks for the footprint that would have been left by the key's insertion:

```
for i = 1 to m do
    if A[h_i(key)] = 0 then
        return "no"
return "yes"
```

   (a) (5 points) Explain why Bloom filter do not suffer from false negatives. (A false negative occurs when a key has been inserted in the Bloom filter, but the LOOKUP returns ’no’.)

   (b) (5 points) Give an example when false positives arise. (A false positive occurs when a key has never been inserted into the Bloom filter, but the LOOKUP returns ’yes’)

   (c) (20 points) Suppose a data set S (containing a total number of $|S|$ keys) is inserted into a Bloom filter that uses m hash functions and a length-n bit array. What is the probability that the $\frac{n}{2}$th bit of the array (assume n is even) is set to 1? Express you answer as a function of n, m and $|S|$ For simplicity, make the following assumptions:

      - For every key k in the data set and hash function h_i of the Bloom filter, $h_i(k)$ is uniformally distributed, with each of the n array positions equally likely.
      - All of the $h_i(k)$'s, ranging over all keys and hash functions, are independent random variables.

Poorly Written Hash Functions [30]

2. In this problem you will look at some hash functions that are designed poorly and think about how they can be broken.

   (a) A novice programmer defines the hash function $h(s) = \sum_{i=0}^{|s|} a_i \pmod{256}$ where a_i is the ASCII value of the i'th character in the string. Give an example of two strings which collide under this hash function. [10]

   (b) This same programmer starts writing better functions, but still makes a grave error. The next hash function hashes integers, and maps an integer x to $x \pmod{2187}$. An unsuspecting user tries to store the sequence of powers of 3 $(1, 3, 9, 27 \ldots)$ in this data structure. How many entries in this sequence can the user store before a collision occurs? What mistake did this programmer make in designing the hash function which allowed this collision to occur? [10]

   (c) This programmer wants to use double hashing to improve query times in the hash table. To that end he implements open addressing with the function $h(k, i) = (h_1(k) + ih_2(k)) \pmod{120}$ where h_1, h_2 are hash functions which output integers. Lets say the user has an unfortunate sequence of inputs $k_1, k_2, \ldots$ such that $h_1(k_1) = h_1(k_2) = h_1(k_3) \cdots = 37$ and $h_2(k_1) = h_2(k_2) = h_2(k_3) \cdots = 24$. How many keys can be inserted into the table before the probing method fails to find an open index? What was the programmer's mistake which allowed the probing method to fail before the table was full? (you can assume that h_1, h_2 are good hash functions, the problem is not the repeated collisions in h_1, h_2, it is the early failure of probing in $h(k, i)$ given those repeated collisions) [10]