SECTION 8: HASHING APPLICATIONS, SET RESEMBLANCE & PRIMALITY TESTING

ISABELLE ZHENG

TABLE OF CONTENTS

BLOOM FILTERS

A **Bloom filter** is a probabilistic data structure used for set membership problems. It is more space efficient than conventional hashing schemes.

- There are m bits and k hash functions $f_1, f_2, \dots f_k$.
- When adding an element x to the set, set bits $f_1(x), f_2(x), \dots f_k(x)$ to 1.
- To check if x is already in the set, check if the corresponding bits are set to 1.

With Bloom filters, we trade away *correctness* for *space* — it's possible we say x is in the Bloom filter when it is not. However, with Bloom filters, we only need m bits of memory.

BLOOM FILTERS EXAMPLE

0	0	0	0	0	0	0

$$S = \{x_1, x_2\}$$

$f_1(x_1) = 1, f_2(x_1) = 4$

$f_1(x_2) = 5, f_2(x_2) = 4$

$f_1(x_3) = 5, f_2(x_3) = 1$

0		0	0			0

FINGERPRINTING

Goal: use a short, identifying "fingerprint" for some pattern P to pattern match in a larger file.

- Hash each set of $|P|$ consecutive characters (sliding window) into a 16 bit value (or other size) by taking mods.
- Randomly select some prime p.
- Instead of taking mods for every set of $|P|$ consecutive characters naively, we can just modify the hashed value for the previous set of $|P|$ characters, which we call N. Let a be the leftmost digit of N and b be the rightmost digit of our new number N'.

$$N' = (10(N - 10^{|P|-1}a) + b) \bmod p$$

When we update, we remove the leftmost digit a and insert a new rightmost digit b.

We can use multiple primes to make the probability of a false positive small.

FINGERPRINTING EXAMPLE

$|P| = 5$

$p = 13$

$N' = \left(10\left(N - 10^{|P-1|}a\right) + b\right) \bmod p$

3	1	4	1	5	2

SET RESEMBLANCE

Our goal: determine whether or not two documents are "near duplicates" – the document similarity problem

How?

- Define set resemblance
- Find a way to estimate resemblance efficiently
- Turn document similarity into a set resemblance problem

DEFINING RESEMBLANCE

Consider two sets *A* and *B*. We define the *resemblance* of *A* and *B* (also called the Jaccard Coefficient) to be:

$$resemblance(A, B) = R(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Notice that:

$$0 \leq R(A, B) \leq 1$$

$R(A, B) = 0$	If two sets are disjoint
$R(A, B) = 1$	If two sets are identical

How long does it take to compare two sets this way?

$O(n^2)$	Naive
$O(n \log n)$	Sort, then compare
$O(n)$	Using hashing

RANDOM PERMUTATIONS

We need a "black box" BB that will efficiently output random permutations on our universe. For example,

$$BB(1, x) = \pi_1(x) \qquad BB(50, x) = \pi_{50}(x)$$

Say our random permutations are in the family: $\pi : [0, 15] \to [0, 15]$. Then, they might look like:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi_1(x)$	9	2	14	11	6	3	7	8	15	10	4	13	12	0	1	5
$\pi_2(x)$	3	4	7	12	6	14	1	5	2	8	15	7	11	13	10	9

We use $\pi_i(A)$ to denote the set of elements obtained by computing $BB(i, x)$ for every x in A ("calling card").

If we have a set $A = \{3, 5, 11, 4\}$ what is $\pi_2(A)$?

ESTIMATING RESEMBLANCE

If we compute $\pi_1(A)$ and $\pi_1(B)$, note that $\min\{\pi_1(A)\} = \min\{\pi_1(B)\}$ only if some element x such that $\pi_1(\mathrm{x}) = \min\{\pi_1(A)\} = \min\{\pi_1(B)\}$.

Then, x, the minimum of the union of two sets $A \cup B$, has to lie in the intersection $A \cap B$.

$$\Pr[\min\{\pi_1(A)\} = \min\{\pi_1(B)\}]$$

$$= \Pr[\min\{\pi_1(A \cup B)\} = \min\{\pi_1(A \cap B)\}]$$

$$= \frac{|A \cap B|}{|A \cup B|} = R(A, B)$$

We can just estimate resemblance by taking many permutations and computing their minimums! Then our estimate for resemblance is just:

$$Estimate\ for\ R(A, B) = \frac{\#\ of\ matches}{\#\ of\ permutations}$$

APPLYING TO DOCUMENT SIMILARITY

We turn **documents** into **sets** using *shingling*, where we hash *k* consecutive words each into a 64 bit (or so) hash value to get a smaller set.

An example of shingling where $k = 5$:

CS 124 is a great class! My favorite part is dynamic programming.

COMPUTING SKETCHES

Then, for each document D, you have a set S_D of shingles. Then, we compute a *sketch* for the document. The sketch of a document, with 100 permutations, would be: $(\min\{\pi_1(S_D)\}, \min\{\pi_2(S_D)\}, \ldots \min\{\pi_{100}(S_D)\})$

Example: Let's say our shingles are $S_D = \{6, 2, 12, 5\}$. What does our sketch look like?

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi_1(x)$	9	2	14	11	6	3	7	8	15	10	4	13	12	0	1	5
$\pi_2(x)$	3	4	7	12	6	14	1	5	2	8	15	7	11	13	10	9

PRIMALITY TESTING

Sometimes we want a prime number p, and sometimes it's so large we can't just check whether it's divisible by 1 to $\sqrt{p}$. So, instead we want efficient algorithms that can tell us if a number is prime.

FERMAT TESTING

Fermat's Little Theorem: If p is prime, and $1 \le a < p$ (note p is not divisible by a), then $a^{p-1} = 1 \, mod \, p$.

For example, if $p = 7$ and $a = 3$, $3^6 = 729 = 1 \, mod \, 7$. (Or, $729 \, mod \, 7 = 1$)

Our test:

1. Given prime candidate n, pick $a < n$.
2. Calculate $a^{n-1} \, mod \, n$.
3. If $a^{n-1} \, mod \, n = 1$ (so $a^{n-1} \, mod \, 1 = n$), then n is an a-pseudoprime. Otherwise, we say n is composite.

(Note we can calculate a^{n-1} efficiently with repeated squaring).

In practice, you will want to try this on many choices of a! However, there are some numbers where n will be an a-pseudoprime for all choices of a. These are called Carmichael numbers, and some examples of them include 561, 1105, and 1729, among infinitely many more.

EXAMPLE OF FERMAT TESTING

$n = 299$
$a = 116$

What about $a = 155$?

RABIN-MILLER TESTING

Our test:

1. Given prime candidate n, let u be such that $n - 1 = 2^i u$.
2. For some a, calculate a^u and its subsequent squares (a^{2u}, a^{4u}, etc.)
3. If at any time we have:
   1. $a^{2^{i-1}u} \neq \pm 1 \; mod \; n$
   2. And $a^{2^i u} = 1 \; mod \; n$

   Then, we have a nontrivial square root of $1 \; mod \; n$ and n must be composite. We call a a "witness" to the compositeness of n.

The Rabin-Miller primality test is very efficient, because if n is composite, a randomly selected a will be a witness with probability at least $\frac{3}{4}$, which means we don't need to check many a's to determine, with high probability, that some n is prime.

EXAMPLE OF RABIN-MILLER TESTING

$$n = 1729$$
$$a = 671$$

PROBLEM 1

Let's explore an application of fingerprinting that checks integer multiplication.

We are given three integers $a, b,$ and c, and we want to determine whether or not $a \cdot b = c$. Suppose that $0 \leq a < 10^{250{,}000}$ and $0 \leq c < 10^{500{,}000}$, so it is not feasible to actually perform the multiplication!

1. Suppose someone told you to check whether 23898239 · 19392981 = 83431298313 is true. How can you tell the answer is *false* immediately?
2. Generalize your strategy to come up with an algorithm that tests whether $a \cdot b = c$. Be sure your algorithm is *randomized* so it works well on average for any a, b, c (hint: choose a prime number!).
3. Using the Prime Number Theorem, which says that there are $\Theta(n/\ln n)$ primes less than n, bound the failure probability of your algorithm, assuming that you randomly choose a prime number below 10^{18}.

PROBLEM 2

Prove that if the resemblance between two documents $R(A, B) = 0$, then our set resemblance algorithm always gives a correct estimate of the resemblance.

PROBLEM 3

Consider the number 1105.

a) Does 1105 pass Fermat's test? (Hint: try $a = 7$ and $a = 5$)

b) Does 1105 pass the Rabin-Miller test?