Randomized Hash and Karp-Rabin Algorithm

Project for CSEP 531 by Stanislav Narivonchik (stanar@uw)

Hash Function

- H(S): where X is a string of any size (n), but H(S) is fixed-size (T)
- Used as checksums, fingerprints, error correction codes
- Hash collision: H(A) = H(B), while A != B.
- How to estimate chance of collision? Need some data model.

Example:

Distribution of a file A, with a checksum. How to estimate transfer error? Need P(B|A), where B is a potential copy. Transmission model? What if there's an adversary willing to fake a copy?

Use Randomized Algorithm

Add randomization independent of data model:

- Choose a uniformly random prime number $p \in \{2, 3, \ldots, T\}$
- Hp(A) = A mod p
- Chance of collision: if A != B, then

$$P(Hp(A) = Hp(B)) < 1.26 (n / \ln n) / (T / \ln T)$$

E.g. if we have T = n * n, then collision probability is O(1/n)

This estimation <u>does NOT depend on the values of A and B</u>.

Proof Sketch

1. Let $\pi(x)$ denote the number of primes <= x.

   For x >=17: $x / \ln x < \pi(x) < 1.26\, x / \ln x$

2. If Hp(A) = Hp(B) then A ≡ B (mod p).

   P(Hp(A) = Hp(B)) <= (# primes dividing |A - B|) / π(T)

3. Number of primes dividing N is less than $\pi(\log 2\, N)$.

   P(Hp(A) = Hp(B)) < π(n) / π(T) < 1.26 (n / ln n) / (T / ln T)

Application: Pattern Matching (Karp-Rabin)

```
function RabinKarp(string s[1..n], string pattern[1..m])
  hpattern := hash(pattern[1..m]);  hs := hash(s[1..m])
  for i from 1 to n-m+1
    if hs = hpattern
      if s[i..i+m-1] = pattern[1..m]
        return i
    hs := hash(s[i+1..i+m])
  return not found
```

Use rolling hash: H(s[i+1..i+m]) = R(H(s[i..i+m-1], s[i], s[i+m]), e.g. Hp(S):

```
Hp := (2 * (Hp - (1<<(m-1)) * s[i]) + s[i+m]) % p;
```

Karp-Rabin Algorithm Modifications

Expected running time is O(n+m). Worst case O(nm).

Need random prime generator – use *Miller-Rabin primality test*.

Modifications:

- Use k different primes, but never check that A=B: worst case O(n+m), the result is not 100%, but anything close enough to 100%.
- Regenerate p, if Hp(A) = Hp(B), but A != B. Hedge against catastrophe (long series of false matches). Expected running time is still O(n+m), if p is not prime!
- Use other rolling hash functions, e.g. *Rabin fingerprint*.

Multiple Pattern Search

Used to detect plagiarism. Expected running time is O(n+k*m).

```
function RabinKarpSet(string s[1..n], set of string subs, m):
    set hsubs := emptySet
    foreach sub in subs
        insert hash(sub[1..m]) into hsubs
    hs := hash(s[1..m])
    for i from 1 to n-m+1
        if hs ∈ hsubs and s[i..i+m-1] ∈ subs
            return i
        hs := hash(s[i+1..i+m])
    return not found
```