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 $A \neq B$.
• 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 $p \in \{2, 3, \ldots, T\}$
• $H_p(A) = A \mod p$
• Chance of collision: if $A \neq B$, then
  
  $$P(H_p(A) = H_p(B)) < \frac{1.26 \left( \frac{n}{\ln n} \right)}{\left( \frac{T}{\ln T} \right)}$$

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

This estimation does NOT depend on the values of $A$ and $B$. 
Proof Sketch

1. Let $\pi(x)$ denote the number of primes $\leq x$.
   For $x \geq 17$: $x / \ln x < \pi(x) < 1.26 x / \ln x$

2. If $H_p(A) = H_p(B)$ then $A \equiv B \pmod{p}$.
   $$P(H_p(A) = H_p(B)) \leq \left( \frac{\text{# primes dividing } |A - B|}{\pi(T)} \right)$$

3. Number of primes dividing $N$ is less than $\pi(\log_2 N)$.
   $$P(H_p(A) = H_p(B)) < \frac{\pi(n)}{\pi(T)} < 1.26 \left( \frac{n}{\ln n} \right) / \left( \frac{T}{\ln T} \right)$$
Application: Pattern Matching (Karp-Rabin)

1 function RabinKarp(string s[1..n], string pattern[1..m])
2   hpattern := hash(pattern[1..m]);  hs := hash(s[1..m])
3   for i from 1 to n-m+1
4     if hs = hpattern
5        if s[i..i+m-1] = pattern[1..m]
6           return i
7     hs := hash(s[i+1..i+m])
8   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 $H_p(A) = H_p(B)$, but $A \neq B$. 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 $O(n+k \cdot m)$.

1 function RabinKarpSet(string s[1..n], set of string subs, m):
2     set hsubs := emptySet
3     foreach sub in subs
4         insert hash(sub[1..m]) into hsubs
5     hs := hash(s[1..m])
6     for i from 1 to n-m+1
7         if hs ∈ hsubs and s[i..i+m-1] ∈ subs
8             return i
9         hs := hash(s[i+1..i+m])
10    return not found