

Rabin Karp Algorithm

- Mihir Dharamshi

- 1. Rabin Karp – Overview
- 2. Basics
- 3. Gruesome Internals ☹
- 4. Applications
- 5. Questions ??

Rabin Karp - Overview

- Naïve string matching algorithm
- Why compare strings as a whole??
- Can we reduce the amount of matching done ??
- Representing strings as digits. Hashing the digits further ...

The Basics

- Every character in the alphabet represented as a digit.
- Assume alphabet set (A, C, T, G, U)
- Represent them as digits (1, 2, 3, 4, 5)
- So, a string ACTGCTGATGG would be
12342341344

- Text $T[1, 2 \dots n]$
- Pattern $P[1, 2, \dots m]$
- Let Pattern P map to decimal 'p'
- Let us create sub strings from the Text of the form
- $T[s+1, \dots, s+m]$ where $s = 0, 1 \dots n-m$
This would map to decimal $t(s)$

We will have a hit when $t(s) == p$

Suppose we are looking for pattern "GCT"
In the text "ACTGCTGATGG"

We have,
Text - ACTGCTGATGG
Pattern - GCT maps to $p = 423$

We match the text 12342341344
... 423 Match after 3 shifts

Complexity ??

Complexity

Processing Time :

- Time to compute $P[1 \dots m] = O(m)$

Matching Time

- Time to compute sub string values

$Ts = O(n - m + 1)$

Total matching time = $O((n-m+1)m)$

Because each substring has m digits ...

So matching m digits each time for $n-m+1$ substrings ..

So, what do we have here ...

Another Naïve ???

Some Facts

- In the worst case, Rabin Karp is no better than a naive string matcher.
- What if we have an immensely large p , then we consequently have large Ts for every case.
- However, highly suited for practical applications where substrings don't differ by a big deal e.g., DNA sequencing
- We shall now see how it performs better for the average case for such applications

Some Improvements

- Let us compute p and all the Ts modulo a suitable modulus q .
eg: $42456 \bmod 37$, $q = 37$

Now, we can compute p modulo q in $O(m)$ time modulo q in $O(n-m+1)$ time.

Now, all we do is match the new digits i.e, p modulo q and Ts modulo q .

How do we save time??

- Note that earlier we were matching each of the m digits in a pattern, so now we are only matching one digit.
- This is the digit representing the sub string.
- So, while processing time is the same $O(m)$
- Matching time reduces from $O((n-m+1)m)$ to $O((n-m+1)1)$
- So, the Total matching time now is $O(m) + O(n-m+1) = O(n)$
- Better than Naïve ??

Potential Problem

- Finding a good hashing function could be a pain. 2 strings could hash to the same value.
- So, we provide an additional check if the hash matches. As you will see in the algorithm, the overhead is not much.
- But, we still need a good hashing function

Some Math for you ☺

- How about the modulus function we used for hashing ?? Can you suggest an optimal one.
- The modulus operator q is chosen such that $10q$ fits into one computer word, so that all necessary computations are done with single precision arithmetic ... Extra floating point arithmetic would have its toll.
- And yeah, q is typically a prime number
- Generalising, we have
$$t(s+1) = (d(t(s) - T[s+1]h) + T[s+m+1]) \bmod q$$

The Algorithm

```
■ RABIN-KARP-MATCHER( T,P, d, q )
1 n = length[ T ];
2 m = length[ P ];
3 h = pow( d, m-1 ) mod q
4 P = 0;
5 t0 = 0;

6 for i = 1 to m (Preprocessing)
7   Do p = (dp + P[ i ]) mod q
8   t0 = (dt0 + T[ i ]) mod q

9 for s = 0 to n-m (Matching)
10  Do if p = ts
11    then if P[ 1...m ] = T[ s+1 .... s+m]
12      then "We have a hit"
13  if s < n-m
14    then t(s+1) = d( ts - T[ s+1 ] h ) + T[ s+m+1 ] mod q
```

Applications

- Molecular Biology - Biological molecules can often be approximated as sequences of nucleotides or amino acids
- Killer App: DNA Sequence matching
- Data analysis – The human genome project.
- Matching Protein Sequences

Questions ???

Thank you

References

- Introduction to algorithms – Cormen, Rivest et al.
- <http://www.eecs.harvard.edu/~ellard/O-97/HTML/root/node43.html>
- <http://www.nist.gov/dads/HTML/karpRabin.html>
- <http://www-igm.univ-mlv.fr/~lecroq/string/examples/exp5.html>
- <http://www-igm.univ-mlv.fr/~lecroq/string/>
- <http://www-igm.univ-mlv.fr/~lecroq/string/node5.html#SECTION0050>