Rabin-Karp Algorithm

CS 491 – Competitive Programming

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Spring 2024

Objectives

- Explain how a *rolling hash* works
- Use a rolling hash to find a pattern in a string quickly

Näive String Matching

A reminder of what not to do....

```
int find(string s, string desire) {
  int found = -1;
  for(int i=0; i<s.length() - desire.length(); ++i) {
    found = 0;
    for(int j=0; j<desire.length(); ++j)
      if (s[i] != s[j])
        break;
      else ++found;
    if (found == desire.length())
      return i;
  } // end for i
  return -1; // not fouund
}
```

- Time complexity is $\mathcal{O}(|P||T|)$

Hash Functions

- Remember hash functions!
  - $h(s)$ should be *fast to compute*
  - $h(s_1) = h(s_2)$ *probably* means $s_1 = s_2$
  - $h(s_1) \neq h(s_2)$ *definitely* means $s_1 \neq s_2$
- Can this help us with string matching?

Hashing

Consider this code

```
int find(string s, string desire) {
  int found = -1;
  for(int i=0; i<s.length() - desire.length(); ++i) {
    if (h(desire) == h(s.substr(i,desire.length())) &&
          desire = s.substr(i,desire.length()))
       return i
  } // end for i
  return -1; // not fouund
}
```

- How about now?

Rolling Hashes

- Consider this hash function:

$$h(c_0 \cdots c_{n-1}) = c_0 a^n + c_1 a^{n-1} + c_2 a^{n-2} + \cdots + c_{n-1} \textit{ modulo } b$$

  - a is a constant (256 is reasonable)
  - b is a large prime number (let's use 100007)
  - c_i is the ith character in a string.
- Try it yourself!
  - Compute the hash for `abc`
  - Compute the hash for `bci`
- Hint: ASCII for a is 95 , `i` is 103

Rolling Hashes, ctd

- Consider this hash function:

$$h(c_0 \cdots c_{n-1}) = c_0 a^n + c_1 a^{n-1} + c_2 a^{n-2} + \cdots + c_{n-1} \; modulo \; b$$

- $h("abc") = 95 \times 256^2 + 96 \times 256 + 97 \; mod \; 100007 = 50159$
- $h("bci") = 96 \times 256^2 + 97 \times 256 + 103 \; mod \; 100007 = 15950$
- Can you convert from one to the other quickly?
  - Add 100007 to prevent "going negative"
  - "Subtract off" a by subtracting $(95 \times 256^2 \; mod \; 100007)$
  - Multiply the remainder by 256 and modulo 100007.
  - Add 103
  - Take the modulus again.
- So: $h("bci") = ((h("abc") + 100007 + (95 \times 256^2 \; mod \; 100007)) \times 256 + 103) \; mod \; 100007$

Setting up

```
int a = 256;
int b = 100007;

int pow = 1;
for(i=1; i<desire.length(); ++i)
  pow = (pow * a) % b;

int hash_s = 0;
int hash_d = 0;

for(i=0; i<desire.length(); ++i) { % assume desire < s
    hash_s = (hash_s * a + s[i]) % b;
    hash_d = (hash_d * a + desire[i]) % b;
}
```

Matching Part

```
while (i < s.length() - desire.length()) {
  if (hash_s == hash_d &&
      desire = s.substr(i,desire.length()))
    return i;
  // Subtract out first letter and shift
  hash_s = ((b + hash_s) - (s[i] * pow % b)) * a % b;
  // Add new letter to end
  hash_s += s[i+desire.length()];
  // Always keep modding!
  hash_s = hash_s % b;
}
return -1; // if failed.
```