COMP550 Assignment 4

Siyang Jing

Given string $s[0,1,...,r-1]$,

denote Ascii: {ASCII characters} $\rightarrow \{0,1,2,...,127\}$,

denote R_{128}: {strings} $\rightarrow$ {radix-128 integers},

then we have:

$R_{128}(s) = \sum_{i=0}^{i=r-1} 128^i \times \text{Ascii}(s[r-1-i])$,

By Horner's Rule, we have:

$R_{128}(s) = ((\text{Ascii}(s[0]) \times 128 + \text{Ascii}(s[1])) \times 128 + ...) \times 128 + \text{Ascii}(s[r-1])$

Written sequentially, we have:

$f_0 = \text{Ascii}(s[0])$

$f_1 = f_0 \times 128 + \text{Ascii}(s[1])$

...

$f_k = f_{k-1} \times 128 + \text{Ascii}(s[k])$

...

$f_{r-1} = f_{r-2} \times 128 + \text{Ascii}(s[r-1])$

$R_{128}(s) = f_{r-1}$

By computation rules of mod, or rules of $\mathbb{Z}_m$, we have:

$g_0 = \text{Ascii}(s[0])\%m$

$g_1 = (g_0 \times 128 + \text{Ascii}(s[1]))\%m$

...

$g_k = (g_{k-1} \times 128 + \text{Ascii}(s[k]))\%m$

...

$g_{r-1} = (g_{r-2} \times 128 + \text{Ascii}(s[r-1]))\%m$

$h(s) = R_{128}(s)\%m = g_{r-1}$

From this, we can develop an iterative method to calculate the hash value. However, we have to pay attention to the details of each step. In calculating $g_k = (g_{k-1} \times 128 + \text{Ascii}(s[k]))\%m$, there are two potential sources of overflow, the multiplication and the addition.

We first calculate $\tilde{g}_k = (g_{k-1} \times 128)\%m$ through multiplication by 2 and take modulo m, repeating 7 times. Since $g_{k-1} < m < 2^{31}$, multiplication by 2 does not cause overflow, and since we modulo m everytime we multiply by 2, the process in all does not overflow.

Then we calculate $g_k = (\tilde{g}_k + \text{Ascii}(s[k]))\%m$. Since $\tilde{g}_k < m < 2^{31}$ and $\text{Ascii}(s[k]) < 128$, the addition does not overflow.

Below are the algorithms.

Algorithm: Times128Modm

Input: q, m

$p \leftarrow q$

for $i = 1$ to 7

$p \leftarrow (p \times 2)\%m$

end

Output: p

This algorithm apparently only takes $O(1)$ space.

Algorithm: StringHashDivisionMethod

Input: string $s[0, 1, ..., r-1]$

$g \leftarrow \text{Ascii}(s[0])\%m$

for $i = 1$ to $r - 1$

$g \leftarrow (\text{Times128Modm}(g, m) + \text{Ascii}(s[i]))\%m$

end

Output: g

This algorithm itself only stores g and thus takes $O(1)$ space. In each iteration, the algorithm calls Times128Modm, and Times128Modm takes $O(1)$ space. The result of Times128Modm is not stored and is used immediately, and therefore in total the algorithm costs $O(1)$ space.