# CS344: Solution to Homework 4

## 1 Problem 1

**Solution.** Let $P$ be a set of $k$ patterns, i.e. $P = \{p_1, \dots, p_k\}$ and $|p_j| = m$ ($j \in [k]$). Given $t[1 \dots n]$, the problem is to find all $i \in [n - m + 1]$ such that $t[i \dots i + m - 1] \in P$.

The fingerprint function of string $s$ is defined as follows:

$$f(s) = \sum_{i=1}^{|s|} s[i] * |\Sigma|^{|s|-i} \bmod q, \quad q \in [1, O(|\Sigma|^5)]$$

which costs $O(|s|)$ time since every element in $s$ has to be encoded.

(1) *Fingerprints of $k$ patterns:* By using above function $f$, it takes $O(m)$ time for each $f(p_i)$ and $O(km)$ in total. Because $m = n/2$, the running time is actually $O(kn)$. Then, we can map these $k$ fingerprints $f(p_1), \dots, f(p_k)$ to a hashtable $T$ of size $O(k)$ which takes $O(k)$ time to build hashtable and O(1) time for each lookup.

(2) *Fingerprints of substrings of $t$:* By using rolling hash (Rabin-Karp) and $f$, it takes $O(m)$ time for $f(t[1, m])$ and $O(n - m)$ for $f(t[i \dots i + m - 1])$ ($i \in [2, n - m + 1]$), so $O(n)$ in total.

(3) *Lookup:* We can query hashtable $T$ to check if $f(t[i \dots i + m - 1]) \in T$ ($i \in [1, n - m + 1]$), which takes $O(n - m + 1)$ time.

The algorithm costs $O(kn)$ time. Notice that no matter how you improve lookup, the total time is determined by the time to build fingerprints of $k$ patterns.

## 2 Problem 2

**Solution.** The fingerprint function $f$ is defined the same as above.

(1) *Fingerprints of pattern $p$:* $p$ is a $m \times m$ matrix. For each row $r$ of $p$ denoted as $p[r, :]$, use $f$ to create a fingerprint $p'_r = f(p[r, :])$ ($r = 1, \dots, m$). Then, apply $f$ again on $[p'_1, \dots, p'_m]$ to generate a final fingerprint of matrix $p$. Let's call it $p''$.

(2) *Fingerprints of submatrix of $t$:* Text $t$ is a $n \times n$ matrix. For each row $r$ of $t$, there are $n - m + 1$ subarrays each of size $m$. Apply rolling hash (Rabin-Karp) and $f$ to generate $n - m + 1$ fingerprints denoted as $t'[r, j]$ where $j = 1, \dots, n - m + 1$. Repeating it for all $n$ rows, we get a temporary matrix $t'$ of size $n \times (n - m + 1)$. Similarly, for each column $c$ of $t'$, apply rolling hash to generate $n - m + 1$ fingerprints denoted as $t''[i, c]$ where $i = 1, \dots, n - m + 1$. Repeating it for all $n - m + 1$ columns, we get a new matrix $t''$ of size $(n - m + 1) \times (n - m + 1)$.

We show that pattern $p$ matches $t$ at $[i, j]$ if and only if $p'' == t''[i, j]$, for all $i, j \in [1, \dots, n - m + 1]$. For any submatrix $t_{ij}$ of text t, i.e. $t_{ij} = t[i..i + m - 1, j..j + m - 1]$, we first apply $f$ on each of its rows to generate fingerprints. Since $t_{ij}$ is of size $m \times m$, each row corresponds to exact one fingerprint, i.e. $t'[k, j] = f(t[k, j..j + m - 1])$ for $k = i, \dots, i + m - 1$. Let $t'_{ij} = [t'[i, j], \dots, t'[i + m - 1, j]]^T$, which is a column vector of size $m$. Apply $f$ again on $t'_{ij}$ to generate a new fingerprint $t''_{ij} = f(t'_{ij})$. It is obvious that $t''_{ij}$ is the element of new matrix $t''$ at position $[i, j]$. Therefore, if we want to compare $p$ and $t_{ij}$, it is equivalent to compare $p''$ and $t''[i, j]$ (sufficient). Furthermore, if $p'' == t''[i, j]$, and we already know $t''[i, j]$ is the fingerprint of submatrix $t_{ij}$, so it is equivalent to the fact $p$ matches $t_{ij}$ (necessary).

*Running time:* The running time to generate every $p'_r$ or $p''$ is $O(m)$ and $O(m(m+1)) = O(n^2)$ to repeat for $m+1$ times. For matrix $t$, it costs $O(n)$ times to compute $n-m+1$ fingerprints for each row and $O(n^2)$ for all $n$ rows, i.e. we get matrix $t'$ in $O(n^2)$ time. Similarly, it costs $O(n(n-m+1)) = O(n^2)$ time to get matrix $t''$. For lookup, it takes $O(1)$ to check $p'' == t''[i,j]$ and repeat it for $(n-m+1)^2 = O(n^2)$ times. Combining everything together, the algorithm takes $O(n^2)$ time.