Cairo University
Faculty of Computers and Artificial Intelligence
Computer Science Department

Biological Sequence Analysis **Karp-Rabin Algorithm** Dr. Amin Allam

1 Mapping strings to integers

Let the number of possible characters in a given alphabet be k. Let $I(a)$ be the order of character a in the alphabet such that the range of $I()$ is $0 \dots k-1$. A string $s_{m-1}s_{m-2}\dots s_1 s_0$ from that alphabet can be mapped to the integer $I(s_0) \times k^0 + I(s_1) \times k^1 + \cdots + I(s_{m-1}) \times k^{m-1}$. A more efficient representation: $(((I(s_{m-1}) \times k + I(s_{m-2})) \times k + I(s_{m-3})) \times k + \cdots + I(s_0))$. This is a kind of one-to-one mapping, where each string corresponds to exactly one integer.

For example, suppose the alphabet is $\{\texttt{0}, \texttt{1}, \dots, \texttt{9}\}$, with $k = 10$.
The orders of characters are: $I(\texttt{0}) = 0, I(\texttt{1}) = 1, \dots, I(\texttt{9}) = 9$.
String 957342 is mapped to integer $2\times10^0+4\times10^1+3\times10^2+7\times10^3+5\times10^4+9\times10^5 = 957342$.
Another method from left to right: $(((((9\times10+5)\times10+7)\times10+3)\times10+4)\times10+2) = 957342$.
String 573426 is mapped to integer $6\times10^0+2\times10^1+4\times10^2+3\times10^3+7\times10^4+5\times10^5 = 573426$.
String 734268 is mapped to integer $8\times10^0+6\times10^1+2\times10^2+4\times10^3+3\times10^4+7\times10^5 = 734268$.

For example, suppose the alphabet is $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$, with $k = 4$.
The orders of characters are: $I(\texttt{A}) = 0, I(\texttt{C}) = 1, I(\texttt{G}) = 2, I(\texttt{T}) = 3$.
String GGTAC is mapped to integer $I(\texttt{C})\times4^0+I(\texttt{A})\times4^1+I(\texttt{T})\times4^2+I(\texttt{G})\times4^3+I(\texttt{G})\times4^4 = 689$.
Another method from left to right: $((((I(\texttt{G})\times k+I(\texttt{G}))\times k+I(\texttt{T}))\times k+I(\texttt{A}))\times k+I(\texttt{C})) = 689$
String GTACT is mapped to integer $3 \times 4^0 + 1 \times 4^1 + 0 \times 4^2 + 3 \times 4^3 + 2 \times 4^4 = 711$.
String TACTC is mapped to integer $1 \times 4^0 + 3 \times 4^1 + 1 \times 4^2 + 0 \times 4^3 + 3 \times 4^4 = 797$.

2 Mapping sub-strings of size m to integers

Suppose the alphabet is $\{\texttt{0}, \texttt{1}, \dots, \texttt{9}\}$, with $k = 10$ and we are interested to find the integer mapping of all sub-strings of size $m = 6$ of the string 95734268.

We can map the first sub-string of size $m = 6$, which is 957342, using the method of the previous section, to integer $2 \times 10^0 + 4 \times 10^1 + 3 \times 10^2 + 7 \times 10^3 + 5 \times 10^4 + 9 \times 10^5 = 957342$. More efficiently, we can compute as $(((((9 \times 10 + 5) \times 10 + 7) \times 10 + 3) \times 10 + 4) \times 10 + 2) = 957342$.

Instead of computing the integer mapping of the second sub-string of size $m = 6$ from scratch using the same method, we can use the integer mapping of the first sub-string of size $m = 6$ which we have already computed. That is, the integer mapping of string 573426 can be computed by subtracting 9×10^5 from the integer mapping of 957342 then multiplying 10 then adding 6. That is, the integer mapping of 573426 is $((957342 - (9 \times 10^5)) \times 10 + 6) = 573426$. Similarly, the integer mapping of string 734268 can be computed by $((573426-(5\times10^5))\times10+8) = 734268$.

Consider another example: suppose the alphabet is $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, with $k = 4$ and we are interested to find the integer mapping of all sub-strings of size $m = 5$ of the string $\mathtt{GGTACTC}$.

We can map the first sub-string of size $m = 5$, which is $\mathtt{GGTAC}$, using the method of the previous section, to integer $((((I(\mathtt{G}) \times k + I(\mathtt{G})) \times k + I(\mathtt{T})) \times k + I(\mathtt{A})) \times k + I(\mathtt{C})) = 689$.

The integer mapping of string $\mathtt{GTACT}$ is $((689 - (I(\mathtt{G}) \times k^{m-1})) \times k + I(\mathtt{T})) = 711$.
The integer mapping of string $\mathtt{TACTC}$ is $((711 - (I(\mathtt{G}) \times k^{m-1})) \times k + I(\mathtt{C})) = 797$.

3 Karp-Rabin algorithm

In order to search for a pattern p of size m inside a string t of size n, we can compute the integer mapping of p, then we compute the integer mapping of all sub-strings of size m inside t using the method in the previous section. Whenever one of the integer mappings of the sub-strings of t matches the integer mapping of p, an occurrence of p is reported in that location in t. We can obtain the integer mapping of a sub-string of t using the integer mapping of the previous sub-string of t by subtracting $I(a)$ where a is the first character of the previous sub-string, and then multiplying by k^{m-1} then adding $I(b)$ where b is the last character of the current sub-string. By pre-computing k^{m-1}, these operations can be done in $O(1)$ time. Therefore, the total time complexity is $O(n+m)$.

When the pattern size m is large, integer values become big and require using big-integer libraries to avoid overflow, which will increase time complexity. Instead, we can use the modular arithmetic operations to avoid overflow. First, we choose a prime number q which is large enough, but should not exceed the square root of the largest number that can result from integer multiplication and does not overflow within the processor. We can replace all arithmetic operations in the previous computations by modular arithmetic operations modulo q. The only difference is that, it is no more one-to-one mapping. That is, several strings can be mapped to the same integer. The effect is that it is possible to find false positives. Therefore, when we find an integer mapping of a sub-string of t that matches the integer mapping p, we must make sure that p exists in that location by character-by-character comparison. This increases the complexity by $m \times (occ + fp)$ where occ is the number of occurrences of p inside t and fp is the number of false positives. The number of false positives can be reduced significantly when q is a large prime.

More formally, suppose we need to compute the integer mapping IC of the string yb using the integer mapping IP of the string ay, where a is the first character of the previous sub-string of size m, b is the last character of the current sub-string of size m, and y is a common sub-string of size $m-1$. Suppose the alphabet size is k.

Without using modular arithmetic, we compute: $IC = (IP - I(a) \times k^{m-1}) \times k + I(b)$
Using modular arithmetic, let q be prime: $IC = ((IP - I(a) \times k^{m-1}) \times k + I(b)) \mod q$
Equivalently: $IC = (((IP - (I(a) \times (k^{m-1} \mod q)) \mod q) \times k) \mod q + I(b)) \mod q$

That is, we perform each operation modulo q to keep the result $< q$ to avoid overflow. If a negative value is encountered, we add q to make it positive, since $((a + q) \mod q) = (a \mod q)$.

The above result follows from the following rule ($*$ can be $\times$, $+$, or $-$):
$((a * b) \mod q) = ((a \mod q) * (b \mod q)) \mod q$

Suppose the alphabet is $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$, with $k = 4$ and we need to search for the pattern $p =$GTACT of size $m = 5$ inside the string GGTACTC using the Karp-Rabin algorithm with $q = 11$. We use a small value of q in this example just for illustration, but practically it should be large. Note that in the following computations, the largest intermediate integer value must be less than q^2.

Initially, we pre-compute $(k^{m-1} \mod q)$ as:
$v = k \mod q = 4 \mod 11 = 4$
$v = (v \times (k \mod q)) \mod q = (4 \times (4 \mod 11)) \mod 11 = 5$
$v = (v \times (k \mod q)) \mod q = (5 \times (4 \mod 11)) \mod 11 = 9$
$v = (v \times (k \mod q)) \mod q = (9 \times (4 \mod 11)) \mod 11 = 3$

Then we compute the integer mapping of the pattern p =GTACT Using the method of the first section with the addition of modulo q after each operation:
$v = I(\texttt{G}) \mod q = 2$
$v = (((v \times k) \mod q) + I(\texttt{T})) \mod q = (((2 \times 4) \mod 11) + 3) \mod 11 = 0$
$v = (((v \times k) \mod q) + I(\texttt{A})) \mod q = (((0 \times 4) \mod 11) + 0) \mod 11 = 0$
$v = (((v \times k) \mod q) + I(\texttt{C})) \mod q = (((0 \times 4) \mod 11) + 1) \mod 11 = 1$
$v = (((v \times k) \mod q) + I(\texttt{T})) \mod q = (((1 \times 4) \mod 11) + 3) \mod 11 = 7$

Similarly, we map the first sub-string of size $m = 5$ of the string GGTACTC, which is GGTAC, to integer:
$v = I(\texttt{G}) \mod q = 2$
$v = (((v \times k) \mod q) + I(\texttt{G})) \mod q = (((2 \times 4) \mod 11) + 2) \mod 11 = 10$
$v = (((v \times k) \mod q) + I(\texttt{T})) \mod q = (((10 \times 4) \mod 11) + 3) \mod 11 = 10$
$v = (((v \times k) \mod q) + I(\texttt{A})) \mod q = (((10 \times 4) \mod 11) + 0) \mod 11 = 7$
$v = (((v \times k) \mod q) + I(\texttt{C})) \mod q = (((7 \times 4) \mod 11) + 1) \mod 11 = 7$
Since 7 equals to the integer mapping of the pattern p, we compare the strings character-by-character which results in a mismatch, meaning that it is a false positive. The pattern does not exist at position 0 of the string.

Next, the integer mapping of string GTACT can be computed as:
$v = (v - (I(\texttt{G}) \times (k^{m-1} \mod q)) \mod q) \mod q = (7 - (2 \times 3) \mod 11) \mod 11 = 1$
$v = ((v \times k) \mod q + I(\texttt{T})) \mod q = ((1 \times 4) \mod 11 + 3) \mod 11 = 7$
Since 7 equals to the integer mapping of the pattern p, we compare the strings character-by-character which results in a match, meaning that it is a true positive. So, we report the occurrence of p at position 1 of the string.

Next, the integer mapping of string TACTC can be computed as:
$v = (v - (I(\texttt{G}) \times (k^{m-1} \mod q)) \mod q) \mod q = (7 - (2 \times 3) \mod 11) \mod 11 = 1$
$v = ((v \times k) \mod q + I(\texttt{C})) \mod q = ((1 \times 4) \mod 11 + 1) \mod 11 = 5$

Since 5 does not equal to the integer mapping of the pattern p, we are sure that the pattern can never exist at position 2 of the string because the algorithm never results in a false negative. It is a true negative.