# Hands-On 1: Algorithm Design 23/24

Filippo Boni Alessio Duè Giacomo Trapani

17 March 2024

## 1 Exercise 1

*Describe and analyze how to transform the randomized Karp-Rabin string search algorithm from Monte Carlo to Las Vegas.*

## Solution

The main idea here is to consider the rolling hash technique, which in the standard implementation is a Monte Carlo algorithm (which may return false positives). Assuming that:

$$P \text{ is a pattern } P : [0, \dots, m-1]$$
$$T \text{ is a text } T : [0, \dots, n-1]$$

### 1.1 Algorithmic Implementation

A possible solution for the problem is to analyze algorithmically a version of the Karp-Rabin method that satisfies the given condition of no errors; here is the proposed approach:

**Lemma 1.** *Every positive integer a has at most* $\log_2 x$ *distinct prime divisors.*

*Proof.* By the *fundamental theorem of arithmetic*, we can write $x$ as a product of primes $X = \prod_{i=1}^{k} p_i$; since $p_1 = 2$, then $x = \prod_{i=1}^{k} p_i \geq 2^k$. Therefore, it follows that $k \leq \log_2 x$. □

**Lemma 2.** *Let* $\pi(x)$ *be the cardinality of the set of prime numbers in the interval* $[2; x]$. *The value of* $\pi(x)$ *for* $x \geq 2$ *is approximately* $\frac{x}{\log(x)}$.

We use an operator hash($s$) that computes the hash for a string, and $U(h, x)$ that updates the rolling hash $h$ with a new character $x$.

Here follows the pseudo-code for the Karp-Rabin Las-Vegas:

**Algorithm 1** KARP-RABIN(T, P, hash, p)

```
m ← |P|
n ← |T|
target ← hash(P)
current_hash ← None
for i ∈ 1, ..., n − m do
    if current_hash ≠ None then
        current_hash ← U(current_hash, T[i + m])
    else
        current_hash ← hash(T[i : m])
    end if
    if current_hash = target then
        if Check_String_Equality(T[i : i + m], P) then
            Print("Match found")
            return (i, i + m)
        end if
    end if
end for
return None
```

## 1.2 Algorithm Analysis

### 1.2.1 False Matches

**Theorem 1.** *Assuming $m \geq 1$, $n \geq 2$, $K$ constant, $B$ a suitably large number in $N$, $p$ a randomly chosen prime number in the interval $[2, B]$, Karp-Rabin reports a false match for a given position with a probability of at most $\frac{1}{K}$*

*Proof.* Let:

$$\text{target} = \text{hash}(P)$$

$$T_i = \text{hash}(T[i : i + m]);$$

Define:

- NoMatch to be the set of indices $i$ for which $T_i \neq$ target, i.e., no match with target;
- $x = \prod_{i \in \text{NoMatch}}(|T_i - \text{target}|)$.

We know that $x$ is a positive integer and since $|\text{NoMatch}| \leq n - m$ and $|T_i - \text{Target}| \leq 2^m$, we can say that $x \leq 2^{m(n-m)} \leq 2^{mn}$ . By Lemma 1, we deduce that $x$ has at most $m \cdot n$ prime divisors. Since we have a false match when $T_i \bmod p =$ target mod $p$, then if there were a false match, $p$ would divide $x$.

The following probability results are derived:

$$\Pr(\text{false match}) \leq \Pr(p \text{ divides } x) = \frac{\text{number of prime divisors of } x}{\pi(B)} \leq \frac{mn}{\pi(B)}$$

Choosing $B = K \cdot m \cdot n$, it follows that $\Pr(\text{false match}) = O\left(\frac{1}{K}\right)$. □

### 1.2.2 Complexity Analysis

**Worst case** we have a false match at every position: checking $m$ characters $n$ times $\Rightarrow O(m \cdot n)$.

**Best case** we have a true match in the first position: checking $m$ characters once $\Rightarrow O(m)$.

**Expected** the probability of a false match at each position is $O(1/K)$, so the expected number of false matches is $O(n/K)$. The expected running time is $O(nm/K + n + m)$. We can choose $K$ such that $m/K = c$ with $c$ constant; for instance $K = m$. We then obtain $O(m + n)$.

The running time is not influenced by the value of $K$, except for the generation of the prime $p$, which is an additive logarithmic term in the asymptotic complexity and can thus be ignored.

# 2 Exercise 2

*Show that $h_{ab}(x) = (ax + b \mod p) \mod m$ is 2-independent.*

# Solution

We call $x_1$, $x_2$ two values in $U$ such that $x_1 \neq x_2$. We have to prove that

$$\Pr_{h \in H}[h(x_1) = y_1 \wedge h(x_2) = y_2] \leq \frac{1}{m^2}.$$

First, we consider the linear system

$$\begin{cases} z_1 = ax_1 + b \mod p \\ z_2 = ax_2 + b \mod p \end{cases}$$

Since $p$ is prime, we know from Lagrange's theorem there is only one mapping from pairs $(x_1, x_2)$ to $(z_1, z_2)$, and we have $p \times (p - 1)$ possible pairs of $(z_1, z_2)$.

We notice the number of values in the same residual class modulo $m$ has a lower bound in $\lfloor p/m \rfloor$ and an upper bound in $\lfloor p/m \rfloor + 1$.

We can now give the result

$$\frac{1}{p(p-1)} \times \left(\lfloor \frac{p}{m} \rfloor\right)^2 \leq \Pr_{h \in H}[h(x_1) = y_1 \wedge h(x_2) = y_2] \leq \frac{1}{p(p-1)} \times \left(\lfloor \frac{p}{m} \rfloor + 1\right)^2$$

We can approximate $p(p - 1)$ with $p^2$ and get the upper bound we have in the thesis. □

# 3 Exercise 3

*Consider the deletion in cuckoo hashing: build an example so that the deletion does not produce a feasible graph, meaning that there is no insertion-only sequence that can lead to that graph. Show how to fix this by randomly choosing among $h_1$ and $h_2$ when inserting.*

# Solution

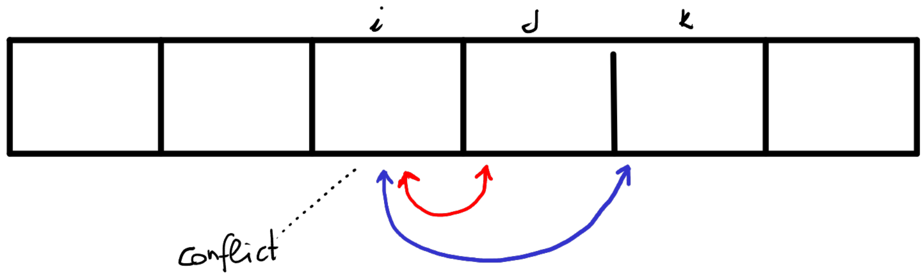
Let's consider a minimal example that shows how deletions can produce a graph which is not obtainable by insertions alone. Let $\mathcal{H} = \left\{h_{a,b} \mid a \in \mathbb{Z}_p^+, b \in \mathbb{Z}_m\right\}$, $h_{a,b}(x) = ((ax+b) \bmod p) \bmod m$, $p = 3$, $m = 2$, $h_1 = h_{1,0} \in \mathcal{H}$, $h_2 = h_{1,1} \in \mathcal{H}$. The hash table has 2 slots and contains value in $\mathbb{Z}_3$. Their hashes are:

| $x$ | 0 | 1 | 2 |
|---|---|---|---|
| $h_1$ | 0 | 1 | 0 |
| $h_2$ | 1 | 0 | 0 |

The sequence of operations `insert 2, insert 0, delete 2` yields the table `[empty, 0]`. This is not obtainable via insertions alone, because if the first position is empty, `0` will be inserted there.

We can reach such a configuration if we randomize the choices between the hash functions. For example, the configuration written above can be reached via insertions with `insert 0` if the hash function $h_2$ can be chosen.

Consider the configuration in the picture below.




Assume red is inserted before blue. We have a conflict for position $i$ and if we deterministically choose the hash function and make a deletion we can get a table which is not achievable via insertions.

However, if we randomize the choice, we can prove there exists a sequence of random insertions s.t. the table we get at the end of the insertions is identical to a certain previously unobtainable configuration.