

Last update: July 26, 2023

Translated

From: e-maxx.ru

# Rabin-Karp Algorithm for string matching

This algorithm is based on the concept of hashing, so if you are not familiar with string hashing, refer to the [string hashing](#) article.

This algorithm was authored by Rabin and Karp in 1987.

**Problem:** Given two strings - a pattern  $s$  and a text  $t$ , determine if the pattern appears in the text and if it does, enumerate all its occurrences in  $O(|s| + |t|)$  time.

**Algorithm:** Calculate the hash for the pattern  $s$ . Calculate hash values for all the prefixes of the text  $t$ . Now, we can compare a substring of length  $|s|$  with  $s$  in constant time using the calculated hashes. So, compare each substring of length  $|s|$  with the pattern. This will take a total of  $O(|t|)$  time. Hence the final complexity of the algorithm is  $O(|t| + |s|)$ :  $O(|s|)$  is required for calculating the hash of the pattern and  $O(|t|)$  for comparing each substring of length  $|s|$  with the pattern.

## Implementation

```
vector<int> rabin_karp(string const& s, string const& t) {
    const int p = 31;
    const int m = 1e9 + 9;
    int S = s.size(), T = t.size();

    vector<long long> p_pow(max(S, T));
    p_pow[0] = 1;
    for (int i = 1; i < (int)p_pow.size(); i++)
        p_pow[i] = (p_pow[i-1] * p) % m;

    vector<long long> h(T + 1, 0);
    for (int i = 0; i < T; i++)
        h[i+1] = (h[i] + (t[i] - 'a' + 1) * p_pow[i]) % m;
    long long h_s = 0;
    for (int i = 0; i < S; i++)
        h_s = (h_s + (s[i] - 'a' + 1) * p_pow[i]) % m;

    vector<int> occurrences;
    for (int i = 0; i + S - 1 < T; i++) {
        long long cur_h = (h[i+S] + m - h[i]) % m;
        if (cur_h == h_s * p_pow[i] % m)
            occurrences.push_back(i);
    }
}
```

```
    return occurrences;  
}
```

## Practice Problems

- [SPOJ - Pattern Find](#)
- [Codeforces - Good Substrings](#)
- [Codeforces - Palindromic characteristics](#)
- [Leetcode - Longest Duplicate Substring](#)

Contributors:

[jakobkogler](#) (50.0%)   [paramsingh](#) (19.64%)   [adamant-pwn](#) (10.71%)   [hieplpvip](#) (8.93%)   [likecs](#) (7.14%)  
[qubits-fan](#) (1.79%)   [joaquingx](#) (1.79%)