PCC104 - Projeto e Análise de Algoritmos

Marco Antonio M. Carvalho

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

Conteúdo

Fonte

Este material é baseado nos livros

- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.
- S. Halim. *Competitive Programming*. 3rd Edition, 2013.
- Ian Parberry and William Gasarch. *Problems on Algorithms*. Second Edition, 2002.
- Ian Parberry *Lecture Notes on Algorithm Analysis and Complexity Theory*. Fourth Edition, 2001.

Licença

Introdução

Programas de edição de texto frequentemente precisam encontrar todas as ocorrências de um padrão em um dado texto;

Tipicamente, o texto é o documento sendo editado (maior), e o padrão procurado é uma palavra específica (menor) fornecida pelo usuário;

Este problema é chamado de **Busca de Padrões**, **Busca em Texto**, **Casamento de Padrões** ou **Casamento de Cadeias de Caracteres** (*String Matching*);

Entre diversas aplicações, algoritmos para este problema são aplicados à busca de padrões em sequências de DNA e na busca por páginas da Internet.

Definições

Seja o texto um arranjo $T[1..n]$ de tamanho n e um padrão $P[1..m]$ de comprimento $m \leq n$;

Os elementos de T e P são caracteres oriundos de um alfabeto finito Σ;

Por exemplo, podemos ter $\Sigma = \{0, 1\}$ ou $\Sigma = \{a, b, c, \ldots, z\}$;

Os arranjos de caracteres T e P são chamados comumente de **cadeias de caracteres** ou *strings*.

Definições

Dizemos que um padrão P ocorre com um **deslocamento** (ou *shift*) s em um texto T se $0 \leq s \leq n - m$ e $T[s + 1..s + m] = P[1..m]$, ou seja $T[s + j] = P[j]$ para $1 \leq j \leq m$;

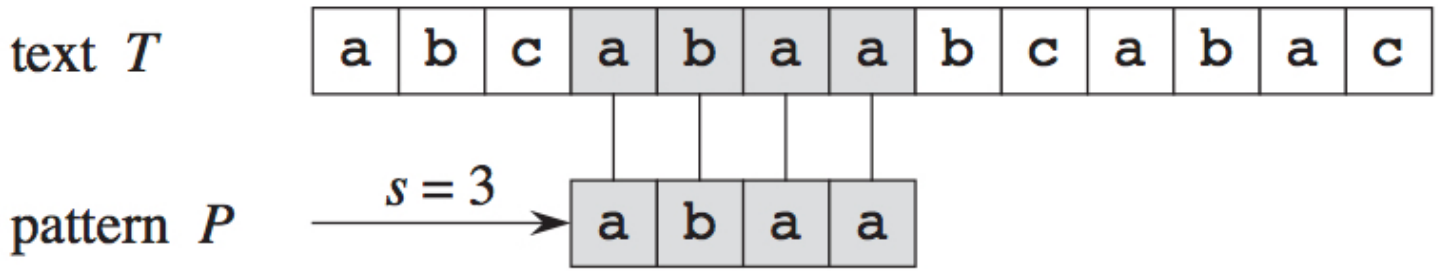
Em outras palavras, podemos dizer que o padrão P ocorre com início na posição $s + 1$ no texto T;

Se o padrão P ocorre no texto T com deslocamento s, então dizemos que s é um deslocamento **válido**;

Caso contrário, s é um deslocamento **inválido**;

O **Problema de Busca de Padrões** nos pede que determinemos todos os deslocamentos válidos nos quais um dado padrão P ocorre em um dado texto T.

Busca de Padrões



Instância do Problema de Busca de Padrões, em que o padrão ocorre apenas uma vez no texto, com deslocamento $s = 3$.

Algoritmos

Veremos três algoritmos para o Problema de Busca de Padrões:

- Um algoritmo ingênuo;
- O algoritmo de Karp-Rabin;
- O algoritmo de Knuth-Morris-Pratt (KMP).

Os dois últimos algoritmos realizam operações de **pré-processamento** baseadas no padrão e então determinam todos os deslocamentos válidos (etapa denominada *matching*);

Desta forma, a complexidade destes algoritmos é analisada somando-se a complexidade das operações de pré-processamento e **matching**;

A seguir, é apresentada a notação e terminologia utilizada pelos algoritmos.

Notação e Terminologia

O conjunto de todas as cadeias de caracteres de comprimento finito formadas sobre o alfabeto Σ é denotado por Σ^*;

A cadeia de caracteres vazia, de comprimento zero, denotada por ϵ, também pertence a Σ^*;

O comprimento de uma cadeia de caracteres x é denotada por $|x|$;

A concatenação de duas cadeias de caracteres x e y é denotada por xy e possui comprimento $|x| + |y|$, consistindo dos caracteres de x seguidos pelos caracteres de y.

Notação e Terminologia

Dizemos que uma cadeia de caracteres w é um **prefixo** da cadeia de caracteres x, denotado por $w \sqsubset x$, se $x = wy$ para alguma cadeia de caracteres $y \in \Sigma^*$;

Analogamente, dizemos que uma cadeia de caracteres w é um **sufixo** da cadeia de caracteres x, denotado por $w \sqsupset x$, se $x = yw$ para alguma cadeia de caracteres $y \in \Sigma^*$;

Por exemplo, temos que `ab` $\sqsubset$ `abcca` e que `cca` $\sqsupset$ `abcca`;

Para quaisquer cadeias de caracteres x e y e um caractere a, temos que $x \sqsupset y$ se e somente se $xa \sqsupset ya$;

Note que ϵ é sufixo e prefixo de todas cadeias de caracteres e que $\sqsubset$ e $\sqsupset$ são relações transitivas.

Algoritmo Ingênuo

Princípio

O algoritmo ingênuo mostrado a seguir determina todos os deslocamentos válidos usando um laço de repetição que verifica se a condição $P[1..m] = T[s+1..s+m]$ para cada um dos $n - m + 1$ possíveis valores de s.

Algoritmo Ingênuo

NaiveStringMatching(T, P)
Entrada: Cadeias de caracteres T e P
$n \leftarrow |T|$;
$m \leftarrow |P|$;
para $s \leftarrow 0$ ***até*** $n - m$ **faça**
  se $P[1..m] = T[s + 1..s + m]$ **então**
    Imprima "O padrão ocorre com deslocamento s";
  fim
fim

Algoritmo Ingênuo

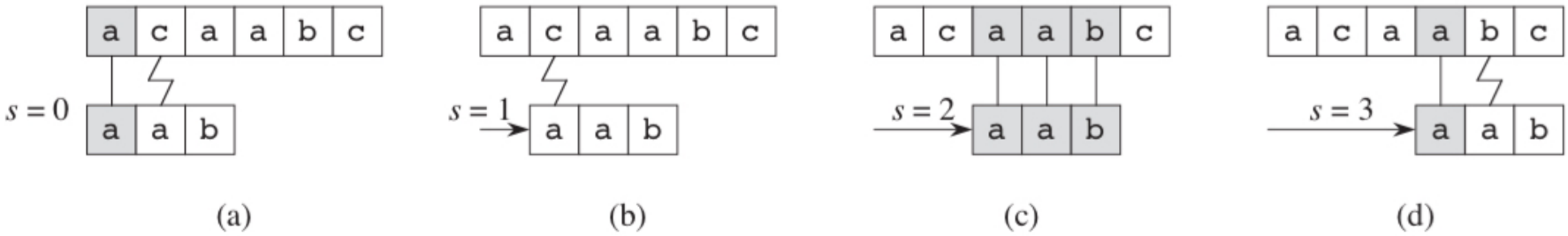
Análise

Neste algoritmo, o padrão é "deslizado" ao longo do texto, verificando-se para quais deslocamentos há uma correspondência total entre os caracteres do padrão e do texto;

O teste da instrução "**se**" verifica implicitamente todos os caracteres para determinar uma correspondência total ou para no primeiro caractere não correspondente ao padrão;

A figura a seguir retrata o algoritmo apresentado.

Algoritmo Ingênuo



Exemplo de execução do algoritmo, em que $P = \texttt{aab}$ e $T = \texttt{acaabc}$.

Algoritmo Ingênuo

Análise

O teste da instrução "**se**" verifica implicitamente todos os caracteres para determinar uma correspondência total ou para no primeiro caractere não correspondente ao padrão;

Este algoritmo possui complexidade $O((n-m+1)m)$, sendo este um limite restrito no pior caso;

Por exemplo, consideremos como texto a cadeia de caracteres a^n (ou seja, uma cadeia composta por n a's) e o padrão a^m;

Para cada um dos $n-m+1$ possíveis deslocamentos, a verificação implícita seria executada m vezes para validar o deslocamento;

Desta forma, nos casos em que $m = \lfloor n/2 \rfloor$, temos que a complexidade é $\Theta(n^2)$.

Algoritmo Ingênuo

Análise

Claramente, o algoritmo ingênuo apresentado não é eficiente para o problema tratado;

A ineficiência deste algoritmo é devida ao mesmo não aproveitar a informação obtida ao computar um valor de s anterior nas computações futuras;

Por exemplo, se $P = \texttt{aaab}$ e determinamos que $s = 0$ é um deslocamento válido, então nenhum dos deslocamentos de valor 1, 2 ou 3, serão válidos, uma vez que $T[4] = \texttt{b}$;

Em seguida, veremos um algoritmo com melhor complexidade no pior caso.

Rabin-Karp

Richard M. Karp (*1935)

- Cientista da Computação americano;
- Prêmio Turing em 1985;
- John von Neumann Theory Prize 1990;
- National Medal of Science 1990;
- Kyoto Prize 2008;
- Famoso por:
  - Lista de 21 problemas NP-Completos;
  - Algoritmo de Edmonds-Karp;
  - Busca em texto;
  - etc...

Rabin-Karp

Michael O. Rabin (*1931)

- Cientista da Computação israelita;
- Prêmio Turing em 1976;
- Prêmio Dijkstra de 2015;
- Famoso por:
  - Teste de primalidade;
  - Criptografia;
  - Autômatos finitos não determinísticos;
  - Algoritmos aleatorizados;
  - Busca em texto;
  - etc...

Introdução

O algoritmo de Rabin-Karp foi proposto em 1987 por Richard M. Karp e Michael O. Rabin;

Este algoritmo possui um bom desempenho na prática, embora sua complexidade no pior caso seja igual ao do algoritmo ingênuo;

É possível generalizar o algoritmo para outros problemas, como busca de padrões bidimensional;

Utiliza noções elementares de teoria dos números, tais como a equivalência de dois números módulo um terceiro número.

Rabin-Karp

Princípio

Suponhamos $\Sigma = \{0, 1, 2, \ldots, 9\}$, de maneira que cada caractere é um dígito (podemos generalizar, supondo que cada caractere é um dígito em notação de base d, em que $d = |\Sigma|$);

Podemos, desta forma, interpretar uma cadeia de k caracteres como um número decimal de k dígitos;

Por exemplo, a cadeia de caracteres `3 1 4 1 5` corresponde ao número decimal 31415;

Dado um padrão $P[1..m]$, seja p o seu valor decimal correspondente;

De maneira semelhante, dado um texto $T[1..n]$, seja t_s o valor decimal associado à subcadeia $T[s+1..s+m]$ (para $s = 0, 1, \ldots, n-m$), cujo comprimento é m;

Certamente, $t_s = p$ se e somente se $T[s+1..s+m] = P[1..m]$, ou seja, s é um deslocamento válido se e somente se $t_s = p$.

Princípio

Se pudermos computar p em tempo $\Theta(m)$ e todos os t_s em tempo $\Theta(n-m+1)$, então poderíamos determinar todos os deslocamentos válidos em tempo $\Theta(m) + \Theta(n-m+1) = \Theta(n)$, comparando p com cada um dos valores t_s;

É possível calcularmos p em tempo $\Theta(m)$ usando a regra de Horner, um algoritmo para computação eficiente de polinômios e mais estável numericamente:

$$p = P[m] + 10(P[m-1] + 10(P[m-2] + \ldots + 10(P[2] + 10P[1])\ldots))$$

Princípio

De maneira similar, podemos calcular t_0 a partir de $T[1..m]$ em tempo $\Theta(m)$;

Para calcularmos os valores restantes $t_1, t_2, \ldots, t_{n-m}$ em tempo $\Theta(n-m)$, basta observar que t_{s+1} pode ser calculado a partir de t_s em tempo constante, pois:

$$t_{s+1} = 10(t_s - 10^{m-1}T[s+1]) + T[s+m+1] \quad (1)$$

- A subtração de $10^{m-1}T[s+1]$ remove o dígito mais significativo de t_s;
- A multiplicação do resultado por 10 desloca o número um dígito à esquerda;
- A adição de $T[s+m+1]$ introduz o próximo dígito menos significativo.

Princípio

Por exemplo, se $m = 5$ e $t_s = 31415$, então desejamos remover o dígito mais significativo $T[s+1] = 3$ e introduzir o próximo dígito menos significativo (suponhamos $T[s+5+1] = 2$):

$$
\begin{aligned}
t_{s+1} &= 10(t_s - 10^{m-1}T[s+1]) + T[s+m+1] \\
t_{s+1} &= 10(31415 - 10000 \times 3) + 2 \\
&= 14152
\end{aligned}
$$

Complexidade

Se pudermos pré-computar a constante 10^{m-1} (é possível fazê-lo em tempo $O(lg\ m)$), então a equação 1 requer um número constante de operações aritméticas;

Desta maneira, podemos computar p em tempo $\Theta(m)$ e podemos computar todos $t_0, t_1, \ldots, t_{n-m}$ em tempo $\Theta(n - m + 1)$;

Em suma, podemos determinar todas as ocorrências do padrão $P[1..m]$ no texto $T[1..n]$ com tempo de pré-processamento $\Theta(m)$ e tempo de *matching* $\Theta(n - m + 1)$ no pior caso.

Particularidade 1

Há uma particularidade no princípio do algoritmo Rabin-Karp: p e t_s podem ser excessivamente grandes para serem manipulados apropriadamente;

Se p possui m caracteres, então não é razoável supor que cada operação aritmética em p leva um tempo "constante";

A solução é computar os valores de p e t_s módulo q, sendo q um valor adequado;

É possível computar $p \ mod \ q$ em tempo $\Theta(m)$, e todos os valores $t_s \ mod \ q$ em tempo $\Theta(n - m + 1)$.

Particularidade 1

Se escolhermos q um número primo tal que $10q$ pode ser armazenado em uma única palavra da arquitetura utilizada, então é possível realizar todas as computações com aritmética de precisão simples;

Em geral, com um alfabeto d-ário $\{0, 1, \ldots, d-1\}$, escolhemos q tal que dq pode ser armazenado em uma palavra e ajustamos a equação 1 para considerar o módulo q:

$$t_{s+1} = (d(t_s - T[s+1]h) + T[s+m+1]) \ mod \ q \tag{2}$$

Na equação, $h \equiv d^{m-1} \ (mod \ q)$ é o valor do primeiro dígito (posição mais significativa) de uma "janela" de texto de m dígitos.

Particularidade 2

A utilização da operação módulo q não é uma situação perfeita: $t_s \equiv p \ (mod \ q)$ não implica em $t_s = p$;

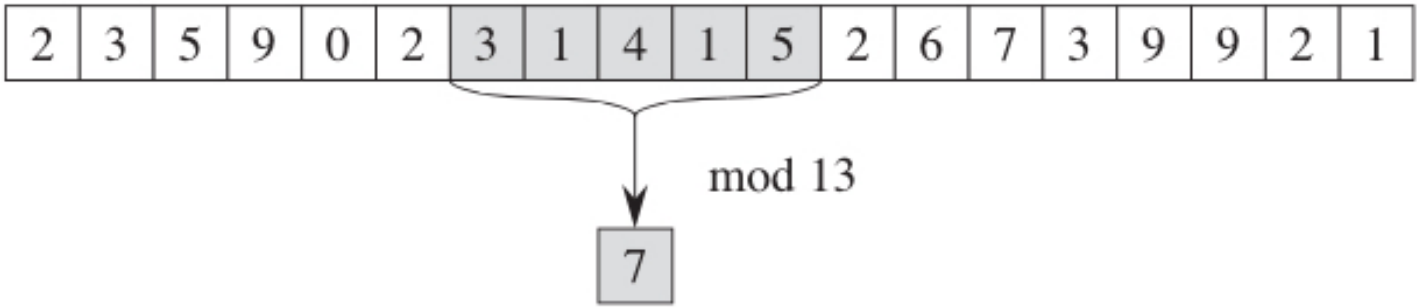
Por outro lado, se $t_s \not\equiv p \ (mod \ q)$ definitivamente implica em $t_s \neq p$, de maneira que o deslocamento s é inválido;

Podemos usar esta propriedade com um regra de teste heurística para deslocamentos inválidos, desde que, para evitarmos falsos positivos, verifiquemos mais profundamente os casos em que $t_s \equiv p \ (mod \ q)$;

Esta verificação adicional explicitamente testa se $P[1..m] = T[s+1..s+m]$;

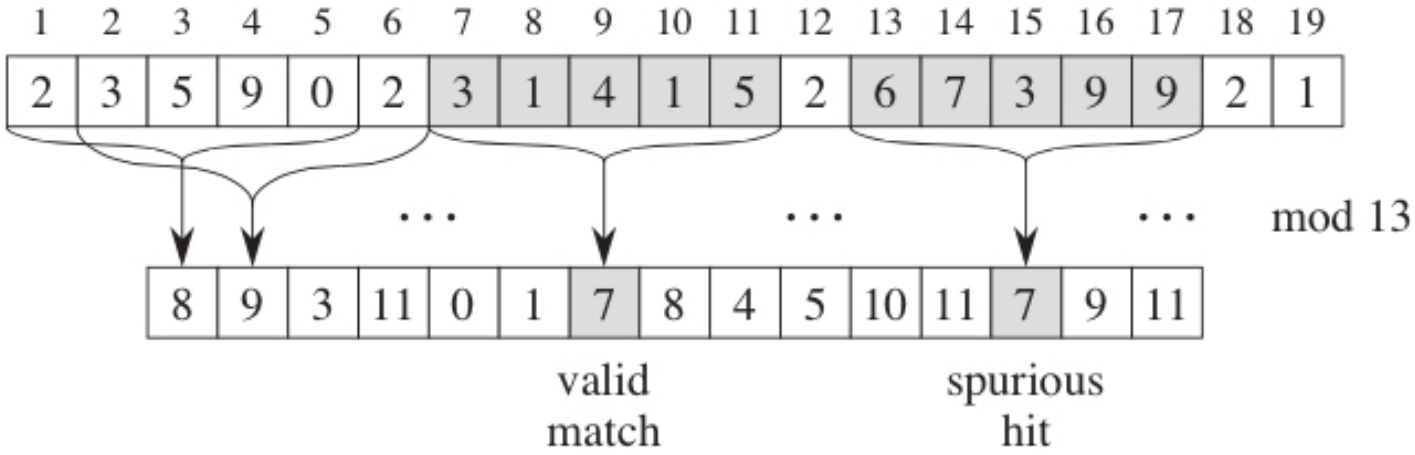
Se q é grande o suficiente, então espera-se que os falsos positivos sejam infrequentes, tal que o custo da verificação adicional seja baixo.

Rabin-Karp



Exemplo de texto em que uma "janela" de comprimento 5 é destacada e cujo valor módulo 13 resulta no valor 7.

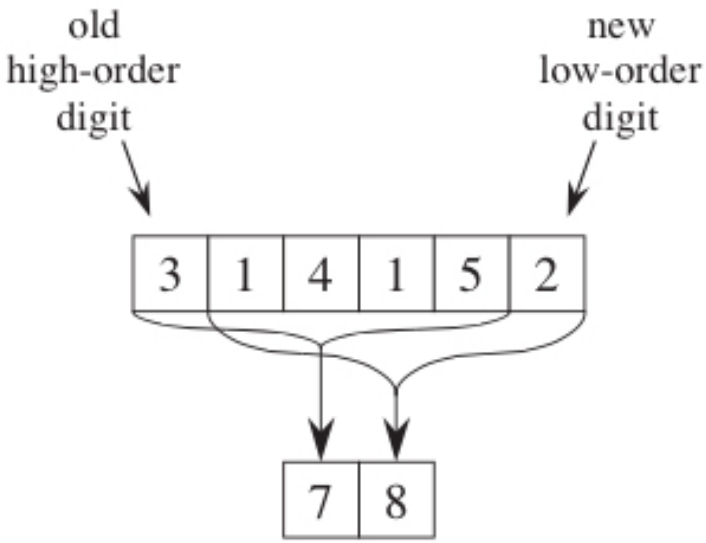
Rabin-Karp

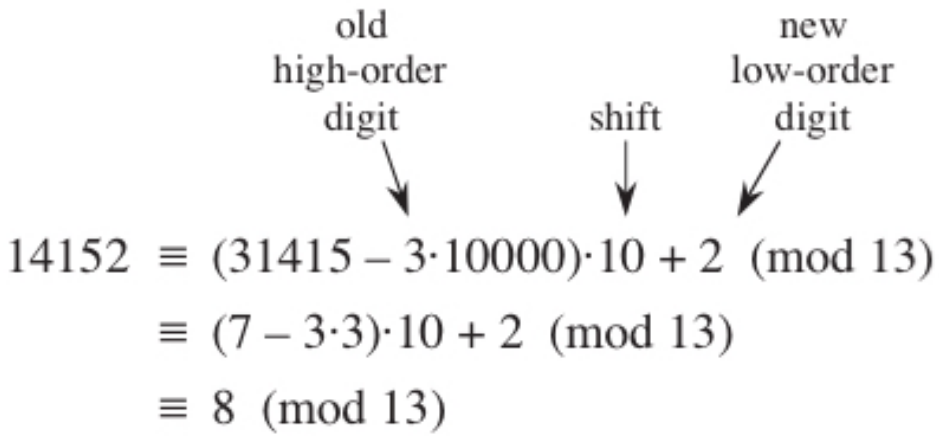


O mesmo texto anterior com valores módulo 13 calculados para cada possível posição da janela de comprimento 5.

Considerando o padrão $P = 31415$, estamos em busca de janelas de texto cujo valor módulo 13 é 7, uma vez que $31415 \equiv 7 \ (mod \ 13)$. O segundo caso é um falso positivo.

Rabin-Karp





Exemplo de cálculo o valor de uma janela de texto em tempo constante, dado o valor da janela anterior.

Rabin-Karp

Karp-Rabin(T, P, d, q)
Entrada: Cadeias de caracteres T e P, base d, número primo q
$n \leftarrow |T|$; $m \leftarrow |P|$; $h \leftarrow d^{m-1} \ mod \ q$; $p \leftarrow 0$; $t_0 \leftarrow 0$;
Pré-processamento
para $i \leftarrow 0$ ***até*** m **faça**
  $p \leftarrow (dp + P[i]) \ mod \ q$;
  $t_0 \leftarrow (dt_0 + T[i]) \ mod \ q$;
fim
Matching
para $s \leftarrow 0$ ***até*** $n - m$ **faça**
  se $p = t_s$ **então**
    se $P[1..m] = T[s + 1..s + m]$ **então**
      Imprima "O padrão ocorre com deslocamento s";
    fim
  fim
  se $s < n - m - 1$ **então**
    $t_{s+1} \leftarrow (d(t_s - T[s + 1]h) + T[s + m + 1] \ mod \ q$
  fim
fim

Observação 1

Novamente, temos que o *matching* do algoritmo Rabin-Karp é $(\Theta(n-m+1)m)$, dado que todos os possíveis deslocamentos válidos são verificados explicitamente, como no caso $T = a^n$ e $P = a^m$;

Porém, na maioria das aplicações práticas, esperamos que as ocorrências do padrão sejam poucas – uma constante c, logo:

$$O((n-m+1)+cm) = O(n+m)$$

Observação 2

É necessário também considerar a verificação dos falsos positivos, que podemos esperar ocorrerem em número $O(n/q)$;

Tendo em vista que há $O(n)$ posições nas quais o teste $p = t_s$ falha e que para verificar cada deslocamento possivelmente válido é exigido tempo $O(m)$, o tempo esperado do *matching* é

$$O(n) + O(m(v + n/q))$$

em que v é o número de deslocamentos válidos e, novamente, n/q é o número de falsos positivos.

Observação 3

Se $v = O(1)$, ou seja, o número de deslocamentos válidos é pequeno, e se escolhermos q como um número primo maior do que o comprimento do padrão, podemos esperar que o *matching* do algoritmo Rabin-Karp exija tempo $O(n + m)$;

Uma vez que $m \leq n$, o tempo de execução esperado é $O(n)$.

Exercício

Considerando $q = 11$, quantos falsos positivos seriam encontrados pelo algoritmo de Rabin-Karp no texto $T = 3141592653589793$ quando buscamos o padrão $P = 26$?

Dúvidas?