Casamento de Cadeias

Introdução

Cadeia de caracteres são definidos por uma sequência de elementos denominados caracteres. Esses caracteres são escolhidos dentro de um conjunto denominado alfabeto, por exemplo, em uma cadeia de bits, o alfabeto é {0,1}. E o objetivo, ou problema, de casamento de cadeias é encontrar todas as ocorrências de um determinado padrão em um texto.

Exemplos de aplicação

- Edição de texto;

- Recuperação de informação;

- Estudo de sequências de DNA em biologia computacional.

Formalização do problema

Para o problema são necessários um texto T e padrão P para a busca/casamento, onde P é necessariamente <= T. Os elementos de T e P são de um alfabeto de tamanho limitado.

T [0...n-1] de tamanho n | P [0...m-1] de tamanho m | m<=n

Alfabeto Σ de tamanho c | exemplo: Σ = {a, b, c, ..., z}

Exemplo: Casamento de cadeias: Dados P[m] e T[n], aonde m<=n, quantas vezes P ocorre em T.

Categorias de Algoritmos

- 1 -P e T não pré-processados;

- 2 -P pré-processado;

- 3 -P e T pré-processados.

1:

Padrão e texto não são conhecidos com antecedência;

Algoritmo sequencial, on-line e de tempo real;

Complexidade de tempo: O (mn);

Complexidade de espaço O (1);

2:

Padrão já conhecido, permitindo seu pré-processamento;

Algoritmo sequencial;

Complexidade de tempo: O (n);

Complexidade de espaço: O (m+c);

Exemplo de aplicação: programas para edição de texto.

3:

Padrão e texto são conhecidos;

Algoritmo constrói índice para texto;

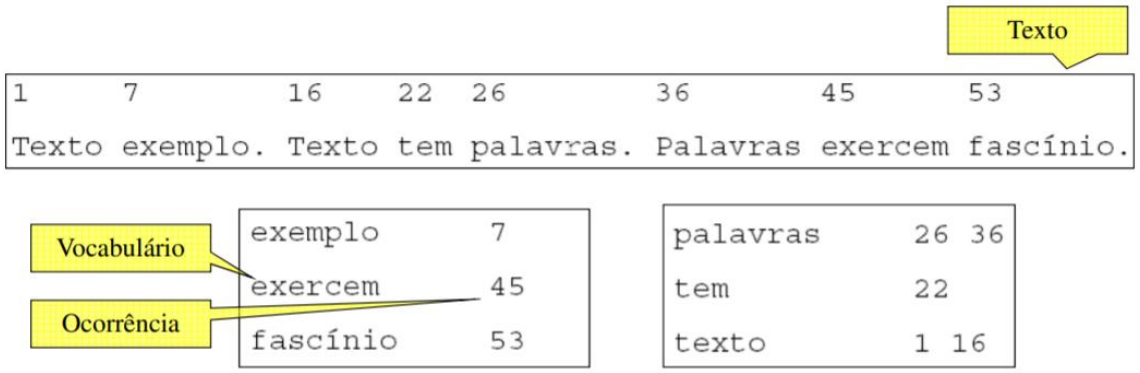
É interessante construir um índice quando a base de dados é grande e semi-estática. O tempo de geração pode ser tão grande quanto O (n) ou O (n log n), mas é compensado por muitas operações de pesquisa no texto. Alguns tipos de índices são: arquivos invertidos, árvore TRIE e árvore PATRICIA e arranjos sufixos.

Complexidade de tempo: O (log n);

Complexidade de espaço: O (n).

Arquivo invertido

Um arquivo invertido é composto por vocabulário e ocorrências, aonde o vocabulário é o conjunto de palavras distintas no texto e para cada palavra distinta, uma lista de posições onde ela ocorre no texto é armazenada, o conjunto de listas é chamada de ocorrência.



A previsão sobre o crescimento do tamanho do vocabulário é definida pela lei de Heaps*.

* **lei Heaps'** (também chamado de **lei de Herdan**) é uma lei empírica que descreve o número de palavras diferentes em um documento (ou conjunto de documentos) em função do comprimento do documento (os chamados relação tipo token). Pode ser formulado como

$$V_R(n) = Kn^{\beta}$$

Aonde K e β dependem das características de cada texto, K geralmente assume valores entre 10 e 100, e β é uma constante entre 0 e 1 (na prática entre 0,4 e 0,6)

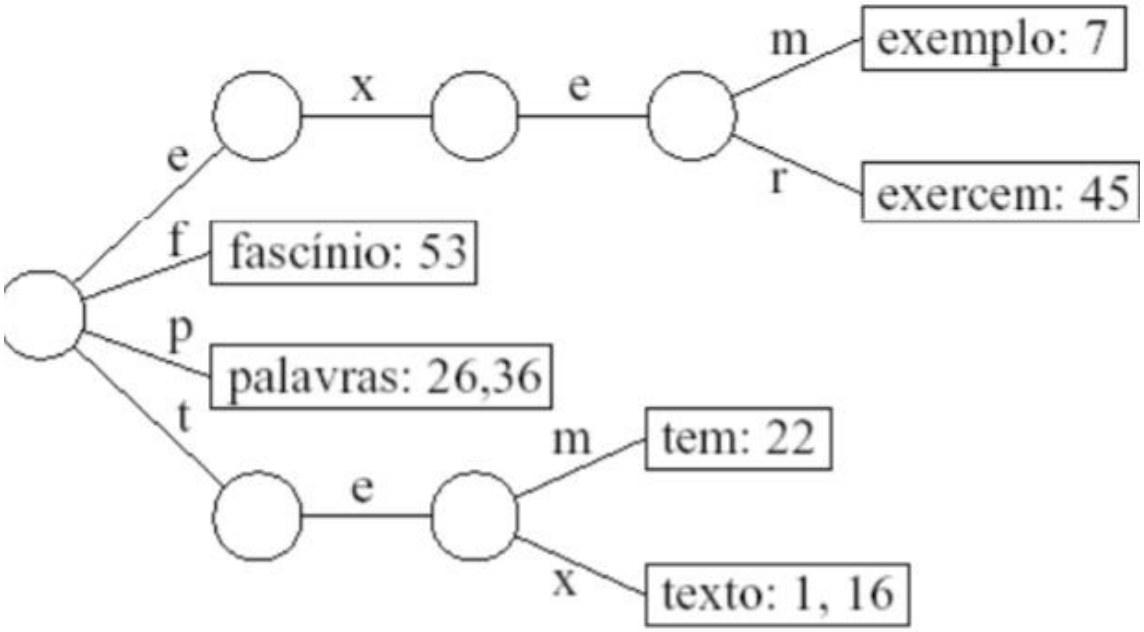
A pesquisa é realizada em 3 passos, pesquisa no vocabulário, recuperação das ocorrências e manipulação das ocorrências. Como a pesquisa de um arquivo invertido sempre começa pelo vocabulário, é interessante

mantê-lo em um arquivo separado, geralmente esse arquivo cabe na memória principal.

A pesquisa por palavras simples pode ser realizada usando qualquer estrutura de dados que torne a pesquisa eficiente, como hashing, árvore TRIE ou árvore B. Aonde as duas primeiras têm custo O(m), onde m é o tamanho da consulta (independentemente do tamanho do texto) e a última possui o custo O (log n); guardar as palavras na ordem lexicográfica é barato em termos de espaço e competitivo em desempenho.

A pesquisa por frases usando índices é mais difícil, já que cada palavra da frase deve ser pesquisada separadamente, no intuito de recuperar listas de ocorrência. Depois, as listas devem ser percorridas de forma sincronizadas para encontrar as ocorrências nas quais as palavras aparecem em ordem.

Exemplo:



O vocabulário do texto é colocando em uma árvore TRIE, armazenando uma lista de ocorrências de cada palavras.

A cada nova palavra lida no texto é feita uma pesquisa na árvore, caso ela já exista, uma nova posição é adicionada na lista de ocorrências, caso o contrário, ela é inserida na árvore e uma nova lista de ocorrências é inicializada com a nova palavra.

Casamento exato

O problema de casamento exato de cadeias consiste em encontrar as ocorrências exatas de um padrão em um texto. Seus algoritmos podem ser categorizados em relação a forma como o padrão é pesquisado no texto.

Leitura dos caracteres do texto T um a um, no intuído de identificar uma ocorrência possível do padrão P. Ex: Força bruta e Shift-And. *1

Pesquisa do padrão P em uma janela que desliza ao longo do texto T, procurando por um sufixo da janela (texto T) que casa com um sufixo de P,

mediante a comparações realizadas da direita para a esquerda. Ex: Boyer-Moore, Boyer-Moore-Horspool e Boyer-Moore-Horspool-Sunday. *2

Autômatos

Um autômato é um modelo de computação muito simples.

Um autômato finito é definido pela tupla (Q, I, F, Σ, T), onde

Q é um conjunto finito de estados;

I é o estado inicial (I ϵ Q);

F é o conjunto de estados finais (F ⊆ Q);

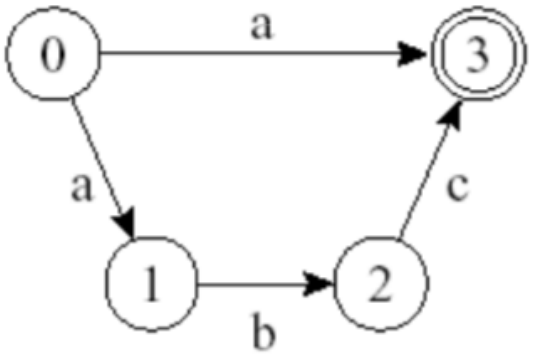
Σ é o alfabeto finito de entrada;

T é a função que define as transições entre os estados*

*T associa a cada estado q ϵ Q um conjunto de estados (q1, q2, ..., qk} ⊆ Q, para cada α ϵ {Σ ∪ {ε}), onde ε é a transição vazia.

Autômato finito não-deterministas

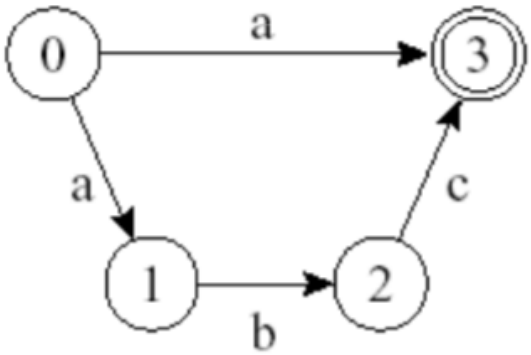
Ocorre quando a função T possibilita a associação de um estado q e um caractere α para mais ude um estado do autômato (ou seja, T (q, α) = {q1, q2, ..., qk} para k>1), ou quando existe alguma transição rotulada por ε.



*No exemplo, a partir do estado 0, por meio do caractere de transição a, é possível atingir os estados 1 e 3.

Autômato finito deterministas

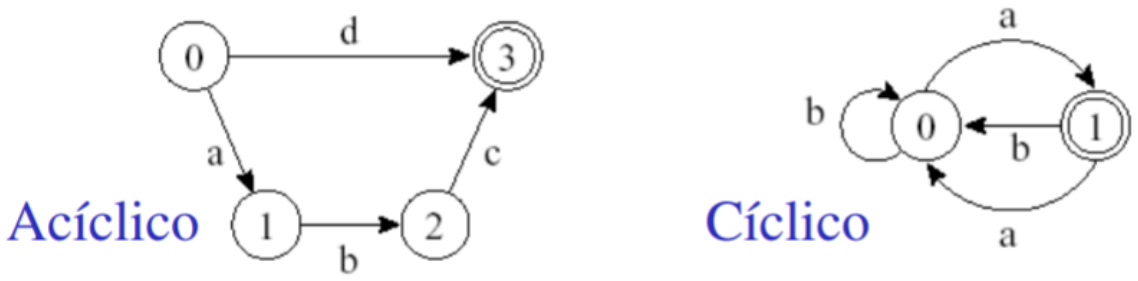
Ocorre quando a função T permite a associação de um estado q e um caractere α para apenas um estado do autômato (ou seja, T (q, α) = {q1}



*Para cada caractere de transição, todos os estados levam a um único estado.

Autômato cíclicos

São aqueles que forma ciclos, isso é se repetem várias vezes dada certa expressão. A linguagem reconhecida por um autômato cíclico pode ser infinita. Por exemplo, o autômato a direita reconhece {ba}, {bba}, {bbba}, {bbbba}, ...



---

*1:

Força bruta

É o algoritmo mais simples para o casamento exato de cadeias, ele consiste em tentar casar qualquer subcadeia de comprimento m no texto com padrão desejado.

Pior caso: C = m x n, ex: P=aab e T=aaaaaaaaaa

Caso esperado:

$$\overline{C_n} = \frac{c}{c-1}\left(1 - \frac{1}{c^m}\right)(n - m + 1) + O(1)$$

*Bem melhor que o pior caso.

Implementação:

```
void ForcaBruta(TipoTexto T, long n, TipoPadrao P, long m)
{ long i, j, k;
  for (i = 1; i <= (n - m + 1); i++)
    { k = i;  j = 1;
      while (T[k-1] == P[j-1] && j <= m) { j++; k++; }
      if (j > m) printf(" Casamento na posicao%3ld\n", i);
    }
}
```

Pesquisa do padrão *P* a partir da *k*-ésima posição do texto *T*

Shif-And Exato

O algoritmo Shift-And usa o conceito de paralelismo de bit, técnica que tira proveito do paralelismo das operações sobre bits dentro de uma palavra de computador, sendo possível empacotar muitos valores em uma única palavra e atualizar todos eles em uma única operação. Uma sequência de bits (b1 ... bc) é chamada de máscara de bits de comprimento c, e é armazenada em alguma posição de uma palavra w do computador.

Algumas operações sobre os bits de uma palavra são:

Repetição de bits: exponenciação (ex.: $01^3 = 0111$);

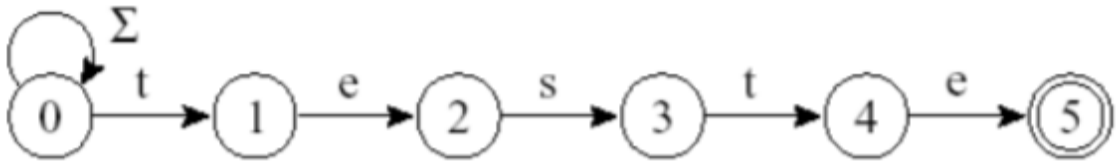
“|”: operador lógico or;

“&”: operador lógico and;

“>>”: operador que move os bits para a direita e entra zeros à esquerda (ex.:b1 b2 ... bc-1 bc >> 2 = 00b1... bc-2).

O algoritmo Shift-And mantém o conjunto de todos os prefixos do padrão P que casam o texto já lido, esse conjunto é representado por uma máscara de bits R = (b1 b2 ... bm); o algoritmo utilza o paralelismo de bits a cada caractere lido do texto.

O algoritmo Shift-And corresponde á simulação de um autômato não-determinista que pesquisa pelo padrão P no texto T. ex.: Autômato que reconhece os prefixos de P = {teste}



Algoritmo shift-and exato

O valor 1 é colocado na j-ésima posição de R (b1 b2 ... bn) se e somente se (p1 ... pj) é um sufixo de (t1 ... ti), onde i corresponde à posição corrente no texto, nesse caso, a j-ésima posição de R é dita estar ativo e o bm átimo significa um casamento exato do padrão.

R1 (novo valor da máscara R) é calculado na leitura do próximo caractere ti+1 do texto. A posição j+1 em R’ ficará ativa se e somente se a posição j estava ativa em R, ou seja, (p1 ... pj) é sufixo de (t1 ... ti), e ti+1 casa com pj+1. *Com o uso de paralelismo de bit, é possível computar a nova máscara com o custo O (1).

Pré-processamento do algoritmo:

Construção de uma tabela M para armazenar uma máscara de bits (b1 b2 ... bm) para cada caractere do padrão P.

Por exemplo, as máscaras de bits para os caracteres presentes em P ={teste} são:

	1	2	3	4	5
M[t]	1	0	0	1	0
M[e]	0	1	0	0	1
M[s]	0	0	1	0	0

*A máscara em M[t] é 10010, pois o caractere t aparece nas posições 1 e 4 do padrão P.

Algoritmo

A máscara de bits R é inicializada como R = 0m

Para cada novo caractere ti+1 lido do texto, o valor da máscara R' é atualizado pela expressão:

$$R' = ((R >> 1) \mid 10^{m-1}) \ \& \ M[T[i]]$$

A operação (R >> 1) desloca as posições para a direita no passo i+1 para manter as posições de P que eram sufixos no passo i.

A operação ((R >> 1) | 10m−1) retrata o fato de que a cadeia vazia ε é também marcada como um sufixo do padrão, permitindo um casamento em qq posição corrente do texto.

Para se manter apenas as posições que ti+1 casa com pj+1, é realizada a conjunção (operador &) entre ((R >> 1) | 10m−1) e a máscara M relativa ao caractere lido do texto.

Exemplo de funcionamento

Pesquisa do padrão P = {teste} no texto T = {os testes ...}.

Texto	$(R >> 1)\|10^{m-1}$					R'				
o	1	0	0	0	0	0	0	0	0	0
s	1	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0
t	1	0	0	0	0	1	0	0	0	0
e	1	1	0	0	0	0	1	0	0	0
s	1	0	1	0	0	0	0	1	0	0
t	1	0	0	1	0	1	0	0	1	0
e	1	1	0	0	1	0	1	0	0	*1*
s	1	0	1	0	0	0	0	1	0	0
	1	0	0	1	0	0	0	0	0	0

Casamento exato

Exemplificação em “código”:

```
Shift–And (P = p1p2 ··· pm, T = t1t2 ··· tn)
{ /*—Préprocessamento—*/
  for (c∈ Σ) M[c] = 0^m;
  for (j = 1; j <= m; j++)  M[pj] = M[pj] |0^(j−1)10^(m−j);
  /*—— Pesquisa ——*/
  R = 0^m;
  for (i = 1; i <= n; i++)
    { R = ((R >> 1|10^(m−1)) & M[T[i]]);
      if ( R & 0^(m−1)1 ≠ 0^m) 'Casamento na posicao i − m + 1';
    }
}
```

Análise:

O custo do algoritmo Shift-And é O (n), desde que as operações sobre os bits possam ser realizadas em O (1) e o padrão caiba em umas poucas palavras do computador.

2*:

Boyer-Moore

O algoritmo clássico Boyer-Moore (BM) surgiu em 1977, e funciona da seguinte forma:

O BM pesquisa o padrão P em uma janela que desliza ao longo do texto T.

Para cada posição desta janela, o algoritmo pesquisa por um 16 sufixo da mesma que casa com um sufixo de P, com comparações realizadas no sentido da direita para a esquerda, se não ocorrer uma desigualdade, uma ocorrência de P em T foi localizada, caso contrário, o algoritmo calcula um deslocamento que a janela deve ser deslizada para a direita antes que uma nova tentativa de casamento se inicie. * O BM propõe duas heurísticas para calcular o deslocamento: ocorrência e casamente.

Algoritmo

A heurística ocorrência alinha o caractere no texto causou a colisão com o 1º caractere no padrão, a esquerda do ponto de colisão, que casa com ele.

Ex.: P = {cacbac}, T = {aabcaccacbac}.

1 2 3 4 5 6 7 8 9 0 1 2

c a c **b** a c – colide em b | c

a a b c a c c a c b a c

c a c b **a** c – colide em a | c

c a c b a **c** – colide em c | a

c a c **b** a c – colide em b | a

c a c b a **c** – colide em c | b

c a c b a **c** – colide em c | a

c a c b a c – Casamento exato

Algoritmo 2

A heurística casamento faz com que, ao mover o padrão a direita, a janela em questão com o pedaço do texto anteriormente casado.

Ex.: P = {cacbac}, T = {aabcaccacbac}.

c a c **b** a c – colide em b | c

a a b c a c c a c b a c

c a c **b** a c – colide em b | a

c a c b a c – Casamento exato

O algoritmo BM decide qual das duas heurísticas deve seguir, escolhendo aquela que provoca o maior deslocamento do padrão, esta escolha implica em realizar comparações para cada colisão que ocorrer, penalizando o desempenho do algoritmo com relação ao tempo de processamento.

Boyer-Moore-Horspool

Em 1980, Horspool apresentou uma simplificação no algoritmo BM, que o tornou mais rápido, ficando conhecido como algoritmo Boyer-Moore-Horspool (BMH). Pela extrema simplicidade de implementação e comprovada eficiência, o BMH deve ser escolhido em aplicações de uso geral que necessitam realizar casamento exato de cadeias.

Simplificação:

Parte da observação de que qualquer caractere já lido do texto a partir do último deslocamento pode ser usado para endereçar uma tabela de deslocamentos, Horspool propôs deslocar a janela de acordo com o valor da tabela de deslocamento relativo ao caractere no texto correspondente ao último caractere padrão.

Para definir a tabela de deslocamentos, faz-se:

- O valor inicial do deslocamento para todos os caracteres do texto é igual a m

- Em seguida, para os m-1 primeiros caracteres do padrão P, os valores do deslocamento são calculados pela regra:

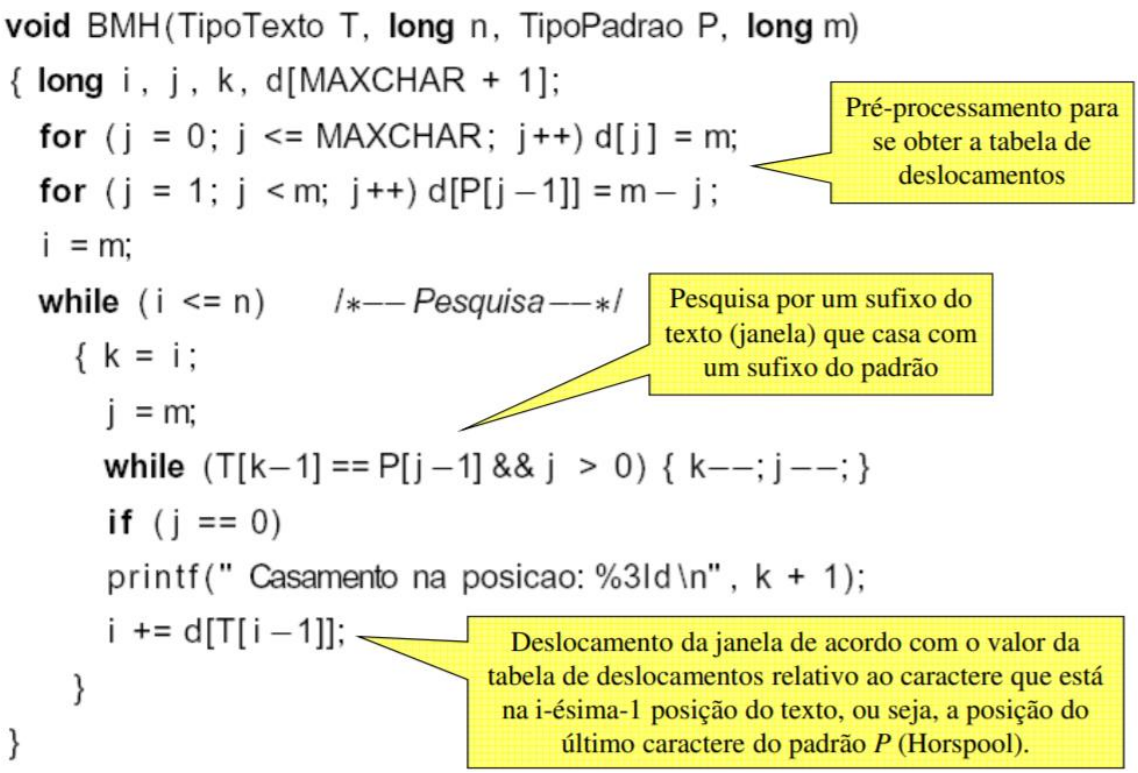
$$d[x] = \min\{j \text{ tal que } (j = m) \mid (1 \leq j < m \ \& \ P[m-j] = x)\}$$

Ex.: Para o padrão P = {teste}, os valores da tabela são:

d[“t”] = 1, d[“e”] = 3, d[“s”] = 2;

d[x] = 5 (valor de m) para todo caractere x do texto que não faça parte do padrão.

Algoritmo/implementação:



Boyer-Moore-Horspool-Sunday

Em 1990, Sunday apresentou uma simplificação importante para o algoritmo BMH, ficando conhecido como algoritmo Boyer-Moore-Horspool-Sunday(BMHS).

Simplificação:

Sunday propôs deslocar a janela de acordo com o valor da tabela de deslocamento relativo ao caractere no texto correspondente ao caractere após o último caractere do padrão.

Para definir a tabela de deslocamentos, faz-se:

O valor inicial do deslocamento para todos os caracteres do texto é igual a m+1;

Em seguida, para os m primeiros caracteres do padrão P, os valores do deslocamento são calculados pela regra:

$d[x] = \min\{j \text{ tal que } (j = m+1) \mid (1 \leq j \leq m \ \& \ P[m+1-j] = x)\}$

Ex.: Para o padrão P = {teste}, os valores da tabela são:

d[“t”] = 2, d[“e”] = 1, d[“s”] = 3;

d[x] = 6 (valor de m+1) para todo caractere x do texto que não faça parte do padrão.

Algoritmo/implementação:

```
void BMHS(TipoTexto T, long n, TipoPadrao P, long m)
{ long i, j, k, d[MAXCHAR + 1];
  for (j = 0; j <= MAXCHAR; j++) d[j] = m + 1;
  for (j = 1; j <= m; j++) d[P[j-1]] = m - j + 1;
  i = m;
  while (i <= n) /*-- Pesquisa --*/
    { k = i;
      j = m;
      while (T[k-1] == P[j-1] && j > 0) { k--; j--; }
      if (j == 0)
      printf(" Casamento na posicao: %3ld\n", k + 1);
      i += d[T[i]];
    }
}
```

Pré-processamento para se obter a tabela de deslocamentos

Pesquisa por um sufixo do texto (janela) que casa com um sufixo do padrão

Deslocamento da janela de acordo com o valor da tabela de deslocamentos relativo ao caractere que está na i-ésima posição do texto, ou seja, a posição seguinte ao último caractere do padrão *P* (Sunday).

Casamento Aproximado

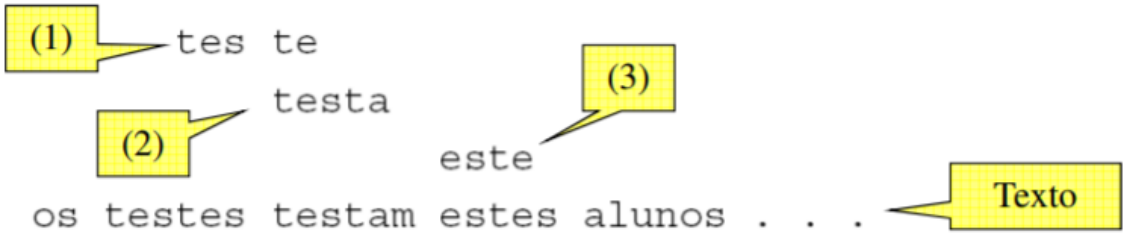
O problema de casamento aproximado de cadeias consiste em encontrar as ocorrências aproximadas de um padrão no texto, logo deve tratar operações de inserção, substituição e retirada de caracteres do padrão.

No exemplo abaixo, aparecem três ocorrências aproximadas do padrão {teste}:

1 Inserção: espaço inserido entre o 3º e 4º caracteres padrão;

2 Substituição: último caractere do padrão substituído pelo a;

3 Retirada: primeiro caractere do padrão retirado.



Distância de edição entre duas cadeias P e P', denota por ed(P, P'), é o menor número de operações necessárias para converter P em P' ou vice-versa. Por exemplo, ed(teste, estende) = 4: valor obtido por meio da retirada do primeiro t de P e a inserção dos caracteres nde ao final de P.

Formalmente, o problema do casamento aproximado de cadeias é o de encontrar todas as ocorrências de P' no texto T tal que ed(P, P') ≤ k, onde k

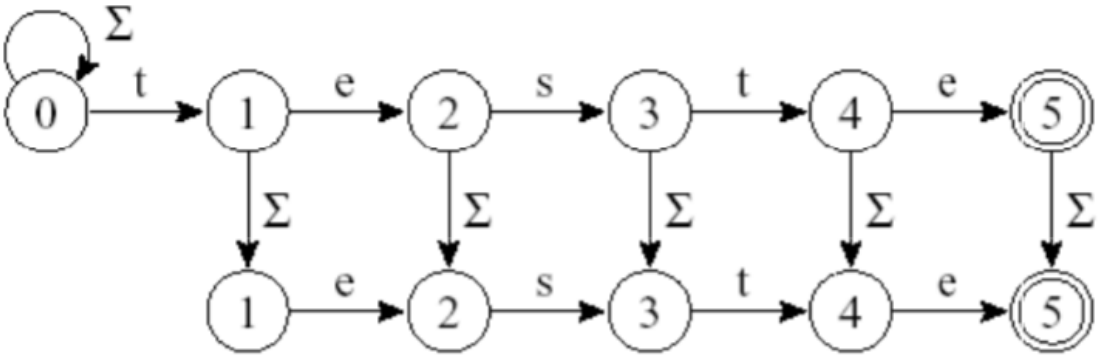
representa o número limite de operações de inserção, substituição e retirada de caracteres necessárias para transformar o padrão P em uma cadeia P'.

O casamento aproximado só faz sentido para 0 < k < m pois, para k = m, toda subcadeia de comprimento m pode ser convertida em P por meio da substituição de m caracteres, k = 0 corresponde ao casamento exato.

Uma medida da fração do padrão que pode ser alterada é dada pelo nível de erro α = k/m. Em geral, α < ½ para a maior parte dos casos.

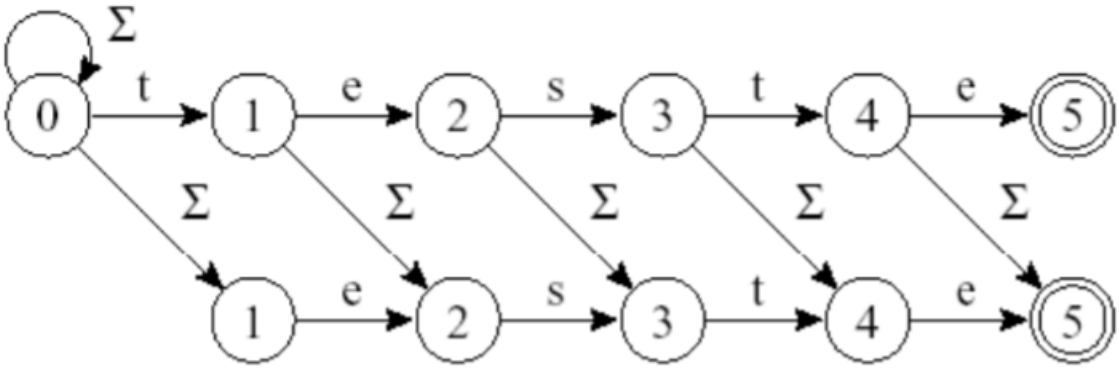
A pesquisa com casamento aproximado é modelada por autômatos não-deterministas.

Autômato que reconhece P ={teste}, permitindo uma inserção:



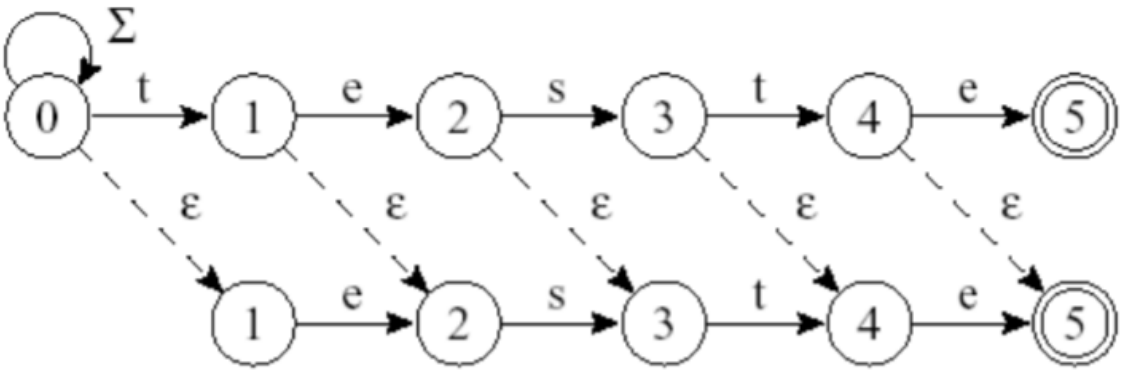
Duas arestas, uma Horizontal, representando o casamento de caractere, avançando-se no texto T e no padrão P, e uma vertical, que insere um caractere no padrão P, avançando-se no texto T mas não no padrão P.

Autômato que reconhece P ={teste}, permitindo uma substituição:



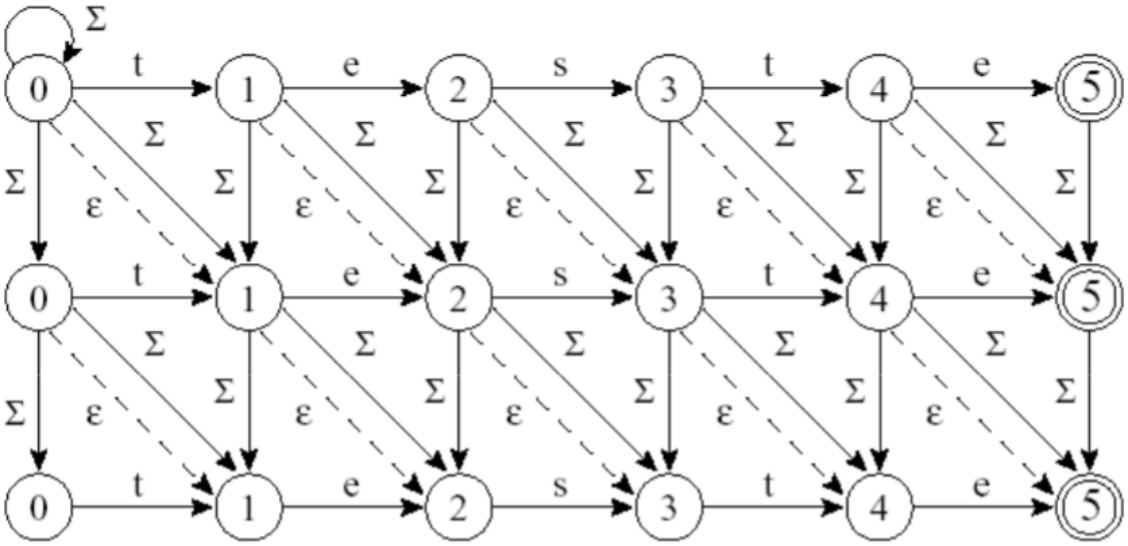
Duas arestas, uma Horizontal, representando o casamento de caractere, avançando-se no texto T e no padrão P, e uma diagonal, que substitui um caractere no padrão P, avançando-se no texto T e no padrão P.

Autômato que reconhece P ={teste}, permitindo uma retirada:



Duas arestas, uma Horizontal, representando o casamento de caractere, avançando-se no texto T e no padrão P, e uma tracejada, que retira um caractere no padrão P, avançando-se no padrão P mas não no texto T (transição vazia).

Casamento Aproximado



O autômato reconhece P ={teste} para k = 2.

Linha 1: casamento exato (k = 0).

Linha 2: casamento aproximado permitindo um erro (k = 1).

Linha 3: casamento aproximado permitindo dois erros (k = 2).

Algoritmo Shift-And Aproximado

```
void Shift-And-Aproximado (P = p1p2...pm, T = t1t2...tn, k)
{ /*-- Préprocessamento --*/
  for (c ∈ Σ) M[c] = 0^m;
  for (j = 1; j <= m; j++)  M[pj] = M[pj] | 0^(j-1)10^(m-j);
  /*-- Pesquisa --*/
  for (j = 0; j <= k; j++) Rj = 1^j 0^(m-j);
  for (i = 1; i <= n; i++)
    { Rant = R0;
      Rnovo = ((Rant >> 1) | 10^(m-1)) & M[T[i]];
      R0 = Rnovo;
      for (j = 1; j <= k; j++)
        { Rnovo = ((Rj >> 1 & M[T[i]]) | Rant | ((Rant | Rnovo) >> 1));
          Rant = Rj;
          Rj = Rnovo | 10^(m-1);
        }
      if (Rnovo & 0^(m-1)1 ≠ 0^m) 'Casamento na posicao i';
    }
}
```

```
void ShiftAndAproximado(TipoTexto T, long n, TipoPadrao P, long m, long k)
{ long Masc[MAXCHAR], i, j, Ri, Rant, Rnovo;
  long R[NUMMAXERROS + 1];
  for (i = 0; i < MAXCHAR; i++) Masc[i] = 0;
  for (i = 1; i <= m; i++) { Masc[P[i-1] + 127] |= 1 << (m - i); }
  R[0] = 0;    Ri = 1 << (m - 1);
  for (j = 1; j <= k; j++) R[j] = (1 << (m - j)) | R[j-1];
  for (i = 0; i < n; i++)
    { Rant = R[0];
      Rnovo = ((((unsigned long)Rant) >> 1) | Ri) & Masc[T[i] + 127];
      R[0] = Rnovo;
      for (j = 1; j <= k; j++)
        { Rnovo = ((((unsigned long)R[j]) >> 1) & Masc[T[i] + 127])
                  | Rant | (((unsigned long)(Rant | Rnovo)) >> 1);
          Rant = R[j];    R[j] = Rnovo | Ri;
        }
      if ((Rnovo & 1) != 0) printf(" Casamento na posicao %12ld\n", i + 1);
    }
}
```