Algoritmos e Estruturas de Dados II

Processamento de Texto

Prof. Tiago Eugenio de Melo
tmelo@uea.edu.br

www.tiagodemelo.info

Observações

- As palavras com a fonte `Courier` indicam as palavras-reservadas da linguagem de programação.

Referências

- **Projetos de Algoritmos – com implementações em Pascal e C**. Nivio Ziviani. 2ª edição. Thomson, 2005.

INTRODUÇÃO

Introdução

Introdução

- O que é casamento de padrões?

Introdução

- O que é casamento de padrões?
  - Consiste em encontrar ocorrências de um certo padrão em uma sequência de elementos.

Introdução

Introdução

- Uma cadeia de caracteres corresponde a uma sequência de elementos denominados caracteres.

Introdução

- Uma cadeia de caracteres corresponde a uma sequência de elementos denominados caracteres.
- Os caracteres são escolhidos de um conjunto denominado alfabeto.

Introdução

- Uma cadeia de caracteres corresponde a uma sequência de elementos denominados caracteres.
- Os caracteres são escolhidos de um conjunto denominado alfabeto.
- Por exemplo, em uma cadeia de bits, o alfabeto é [0, 1].

Introdução

- Uma cadeia de caracteres corresponde a uma sequência de elementos denominados caracteres.
- Os caracteres são escolhidos de um conjunto denominado alfabeto.
- Por exemplo, em uma cadeia de bits, o alfabeto é [0, 1].
- O processamento em cadeias de caracteres é um componente importante em diversos problemas computacionais:

Introdução

- Uma cadeia de caracteres corresponde a uma sequência de elementos denominados caracteres.
- Os caracteres são escolhidos de um conjunto denominado alfabeto.
- Por exemplo, em uma cadeia de bits, o alfabeto é [0, 1].
- O processamento em cadeias de caracteres é um componente importante em diversos problemas computacionais:
  - Edição de texto.

Introdução

- Uma cadeia de caracteres corresponde a uma sequência de elementos denominados caracteres.
- Os caracteres são escolhidos de um conjunto denominado alfabeto.
- Por exemplo, em uma cadeia de bits, o alfabeto é [0, 1].
- O processamento em cadeias de caracteres é um componente importante em diversos problemas computacionais:
  - Edição de texto.
  - Recuperação de informação.

Introdução

- Uma cadeia de caracteres corresponde a uma sequência de elementos denominados caracteres.
- Os caracteres são escolhidos de um conjunto denominado alfabeto.
- Por exemplo, em uma cadeia de bits, o alfabeto é [0, 1].
- O processamento em cadeias de caracteres é um componente importante em diversos problemas computacionais:
  - Edição de texto.
  - Recuperação de informação.
  - Entre outros.

Introdução

Introdução

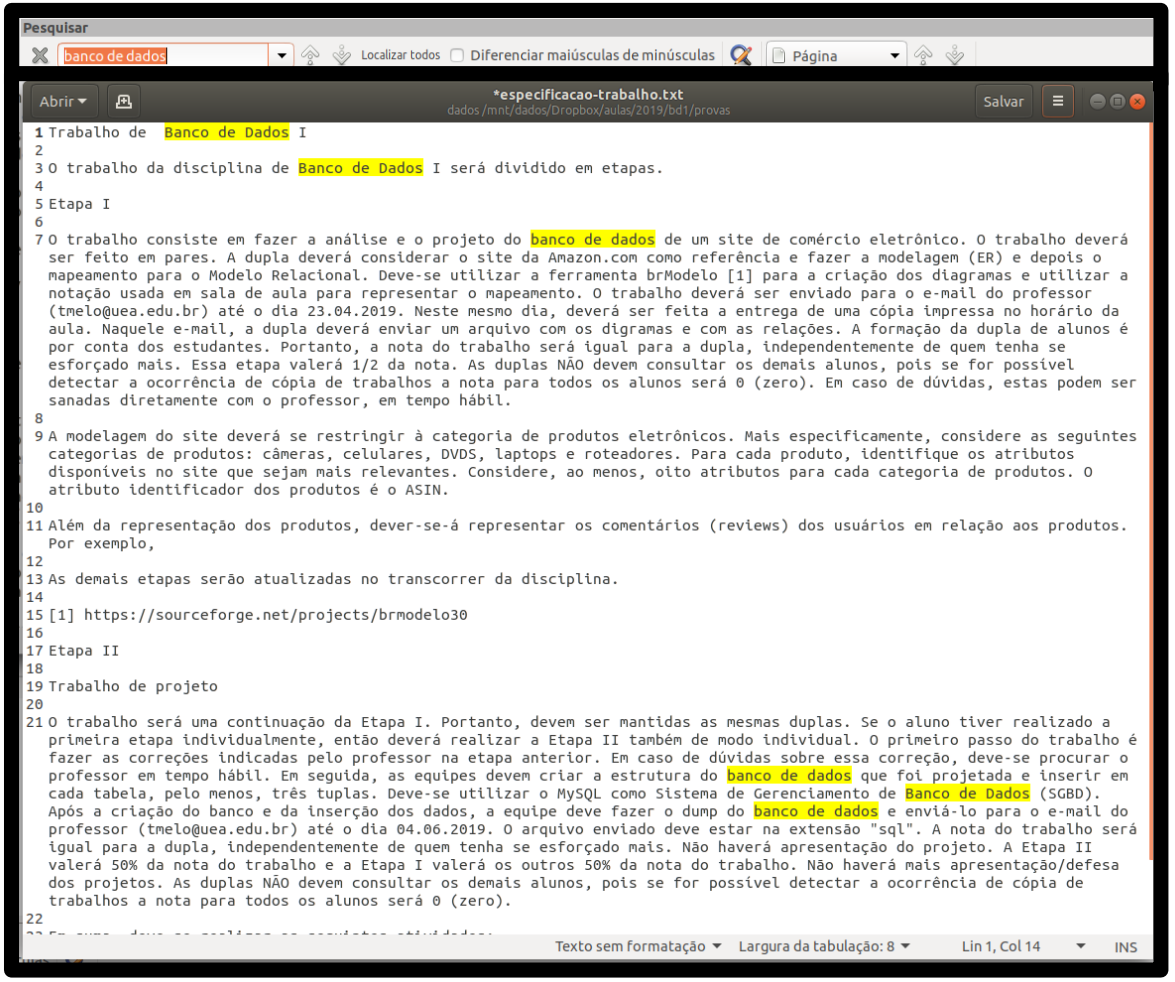
- Em programas editores de texto, o usuário pode querer buscar todas as ocorrências de um determinado padrão (um termo em particular) no texto que está sendo editado.

Introdução

- Exemplo

Introdução

- Exemplo



Introdução

Introdução

- Esse problema é conhecido como casamento de cadeia de caracteres ou casamento de padrão (***pattern matching***).

Introdução

- Esse problema é conhecido como casamento de cadeia de caracteres ou casamento de padrão (***pattern matching***).
- O problema de casamento de cadeias ou casamento de padrões pode ser formalizado.

Introdução

Introdução

- O texto é um arranjo T[1..n] de tamanho n.

Introdução

- O texto é um arranjo T[1..n] de tamanho n.
- O padrão é um arranjo P[1..m] de tamanho m <= n.

Introdução

- O texto é um arranjo T[1..n] de tamanho n.
- O padrão é um arranjo P[1..m] de tamanho m <= n.
- Os elementos de P e T são escolhidos de um alfabeto finito Σ de tamanho c.

Introdução

- O texto é um arranjo T[1..n] de tamanho n.
- O padrão é um arranjo P[1..m] de tamanho m <= n.
- Os elementos de P e T são escolhidos de um alfabeto finito Σ de tamanho c.
- Exemplos de alfabeto:

Introdução

- O texto é um arranjo T[1..n] de tamanho n.
- O padrão é um arranjo P[1..m] de tamanho m <= n.
- Os elementos de P e T são escolhidos de um alfabeto finito Σ de tamanho c.
- Exemplos de alfabeto:

  Σ = {0,1} ou Σ = {a, b, .., z}

Introdução

- O texto é um arranjo T[1..n] de tamanho n.
- O padrão é um arranjo P[1..m] de tamanho m <= n.
- Os elementos de P e T são escolhidos de um alfabeto finito Σ de tamanho c.
- Exemplos de alfabeto:

  Σ = {0,1} ou Σ = {a, b, .., z}
- Dadas duas cadeias P (padrão) de comprimento

Introdução

- O texto é um arranjo T[1..n] de tamanho n.
- O padrão é um arranjo P[1..m] de tamanho m <= n.
- Os elementos de P e T são escolhidos de um alfabeto finito Σ de tamanho c.
- Exemplos de alfabeto:

  Σ = {0,1} ou Σ = {a, b, .., z}
- Dadas duas cadeias P (padrão) de comprimento

  |P| = m e T (texto) de comprimento |T| = n, em que n > m, deseja-se saber as ocorrências de P em T.

Introdução

Introdução

- **`w`** é prefixo de **`x`**, se **`x`** = **`wy`** para alguma cadeia **`y`**. Isso é representado por **`w`** □ **`x`**.

Introdução

- **`w`** é prefixo de **`x`**, se **`x`** = **`wy`** para alguma cadeia **`y`**. Isso é representado por **`w`** □ **`x`**.
- Para representar o sufixo, teremos que **`w`** □ **`x`**.

Introdução

Introdução

- Considerando os dados de entrada como sendo o texto T e o padrão P, temos as seguintes categorias de algoritmos:

Introdução

- Considerando os dados de entrada como sendo o texto T e o padrão P, temos as seguintes categorias de algoritmos:
  - Padrão e texto não são pré-processados.

Introdução

- Considerando os dados de entrada como sendo o texto T e o padrão P, temos as seguintes categorias de algoritmos:
  - Padrão e texto não são pré-processados.
  - Padrão pré-processado.

Introdução

- Considerando os dados de entrada como sendo o texto T e o padrão P, temos as seguintes categorias de algoritmos:
  - Padrão e texto não são pré-processados.
  - Padrão pré-processado.
  - Padrão e texto são pré-processados.

Introdução

Introdução

- Padrão e texto não são pré-processados:

Introdução

- Padrão e texto não são pré-processados:
  - Os algoritmos são do tipo sequencial, on-line e de tempo real, pois tanto o padrão quanto o texto não são conhecidos antecipadamente.

Introdução

- Padrão e texto não são pré-processados:
  - Os algoritmos são do tipo sequencial, on-line e de tempo real, pois tanto o padrão quanto o texto não são conhecidos antecipadamente.
  - Estes algoritmos têm complexidade de tempo O(m.n) para o pior caso.

Introdução

- Padrão e texto não são pré-processados:
  - Os algoritmos são do tipo sequencial, on-line e de tempo real, pois tanto o padrão quanto o texto não são conhecidos antecipadamente.
  - Estes algoritmos têm complexidade de tempo O(m.n) para o pior caso.
  - Um exemplo é o algoritmo **força bruta**.

Introdução

Introdução

- Padrão pré-processado:

Introdução

- Padrão pré-processado:
  - Os algoritmos são do tipo sequencial e o padrão é conhecido anteriormente, o que permite o seu pré-processamento.

Introdução

- Padrão pré-processado:
  - Os algoritmos são do tipo sequencial e o padrão é conhecido anteriormente, o que permite o seu pré-processamento.
  - Estes algoritmos têm complexidade de tempo O(n) para o pior caso.

Introdução

- Padrão pré-processado:
  - Os algoritmos são do tipo sequencial e o padrão é conhecido anteriormente, o que permite o seu pré-processamento.
  - Estes algoritmos têm complexidade de tempo O(n) para o pior caso.
  - Representantes típicos desta categoria são os programas para edição de textos.

Introdução

- Padrão pré-processado:
  - Os algoritmos são do tipo sequencial e o padrão é conhecido anteriormente, o que permite o seu pré-processamento.
  - Estes algoritmos têm complexidade de tempo O(n) para o pior caso.
  - Representantes típicos desta categoria são os programas para edição de textos.
  - Os algoritmos mais conhecidos nesta categoria são Knuth-Morris e Pratt, o Boyer-Moore e o Shif-And.

Introdução

Introdução

- Padrão e texto são pré-processados:

Introdução

- Padrão e texto são pré-processados:
  - Os algoritmos constroem um índice para permitir uma complexidade de tempo O(log n) ou menos.

Introdução

- Padrão e texto são pré-processados:
  - Os algoritmos constroem um índice para permitir uma complexidade de tempo O(log n) ou menos.
  - O tempo de pré-processamento do texto para obter o índice pode ser tão grande quanto O(n) ou O(n.log n), mas esse tempo é compensado por muitas operações de pesquisa no texto.

Introdução

- Padrão e texto são pré-processados:
  - Os algoritmos constroem um índice para permitir uma complexidade de tempo O(log n) ou menos.
  - O tempo de pré-processamento do texto para obter o índice pode ser tão grande quanto O(n) ou O(n.log n), mas esse tempo é compensado por muitas operações de pesquisa no texto.
  - Vale a pena construir um índice quando a base de dados é grande e muda pouco.

Introdução

- Padrão e texto são pré-processados:
  - Os algoritmos constroem um índice para permitir uma complexidade de tempo O(log n) ou menos.
  - O tempo de pré-processamento do texto para obter o índice pode ser tão grande quanto O(n) ou O(n.log n), mas esse tempo é compensado por muitas operações de pesquisa no texto.
  - Vale a pena construir um índice quando a base de dados é grande e muda pouco.
  - Os tipos de índice mais conhecidos são os arquivos invertidos, árvores Trie e Patrícia, e arranjos de sufixos.

CASAMENTO EXATO

Casamento Exato

Casamento Exato

- No casamento de padrão, o problema mais elementar consiste em obter todas as ocorrências exatas do padrão no texto.

Casamento Exato

- No casamento de padrão, o problema mais elementar consiste em obter todas as ocorrências exatas do padrão no texto.
- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques:

Casamento Exato

- No casamento de padrão, o problema mais elementar consiste em obter todas as ocorrências exatas do padrão no texto.
- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques:
  - O primeiro enfoque consiste em ler os caracteres do texto um a um e, a cada passo, algumas variáveis são atualizadas de forma a identificar uma ocorrência possível.

Casamento Exato

- No casamento de padrão, o problema mais elementar consiste em obter todas as ocorrências exatas do padrão no texto.
- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques:
  - O primeiro enfoque consiste em ler os caracteres do texto um a um e, a cada passo, algumas variáveis são atualizadas de forma a identificar uma ocorrência possível.
  - Exemplos de algoritmos:

Casamento Exato

- No casamento de padrão, o problema mais elementar consiste em obter todas as ocorrências exatas do padrão no texto.
- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques:
  - O primeiro enfoque consiste em ler os caracteres do texto um a um e, a cada passo, algumas variáveis são atualizadas de forma a identificar uma ocorrência possível.
  - Exemplos de algoritmos:
    - Força bruta.

Casamento Exato

- No casamento de padrão, o problema mais elementar consiste em obter todas as ocorrências exatas do padrão no texto.
- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques:
  - O primeiro enfoque consiste em ler os caracteres do texto um a um e, a cada passo, algumas variáveis são atualizadas de forma a identificar uma ocorrência possível.
  - Exemplos de algoritmos:
    - Força bruta.
    - Knuth-Morris-Pratta.

Casamento Exato

Casamento Exato

- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques (cont.):

Casamento Exato

- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques (cont.):
  - O segundo enfoque consiste em pesquisar o padrão P em uma janela que desliza ao longo do texto T.

Casamento Exato

- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques (cont.):
  - O segundo enfoque consiste em pesquisar o padrão P em uma janela que desliza ao longo do texto T.
  - Para cada posição desta janela, o algoritmo faz uma pesquisa por um sufixo da janela que casa com um sufixo de P, mediante comparações realizadas no sentido da direita para esquerda.

Casamento Exato

- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques (cont.):
  - O segundo enfoque consiste em pesquisar o padrão P em uma janela que desliza ao longo do texto T.
  - Para cada posição desta janela, o algoritmo faz uma pesquisa por um sufixo da janela que casa com um sufixo de P, mediante comparações realizadas no sentido da direita para esquerda.
  - Exemplo de algoritmo:

Casamento Exato

- Os algoritmos para o casamento exato, em geral, funcionam em dois enfoques (cont.):
  - O segundo enfoque consiste em pesquisar o padrão P em uma janela que desliza ao longo do texto T.
  - Para cada posição desta janela, o algoritmo faz uma pesquisa por um sufixo da janela que casa com um sufixo de P, mediante comparações realizadas no sentido da direita para esquerda.
  - Exemplo de algoritmo:
    - Boyer-Moore.

HISTÓRICO

Histórico

Histórico

- Em 1970, S. Cook provou um resultado teórico sobre um tipo particular de autômato que implicava na existência de um algoritmo de casamento de padrão com tempo proporcional a M + N no pior caso.

Histórico

- Em 1970, S. Cook provou um resultado teórico sobre um tipo particular de autômato que implicava na existência de um algoritmo de casamento de padrão com tempo proporcional a M + N no pior caso.
- Knuth e Pratt, seguindo a construção que Cook usou na demonstração do seu teorema obtiveram um algoritmo relativamente simples e prático.

Histórico

- Em 1970, S. Cook provou um resultado teórico sobre um tipo particular de autômato que implicava na existência de um algoritmo de casamento de padrão com tempo proporcional a M + N no pior caso.
- Knuth e Pratt, seguindo a construção que Cook usou na demonstração do seu teorema obtiveram um algoritmo relativamente simples e prático.
- Ocorreu também que Morris descobriu praticamente o mesmo algoritmo como solução de um problema de edição de texto.

Histórico

- Em 1970, S. Cook provou um resultado teórico sobre um tipo particular de autômato que implicava na existência de um algoritmo de casamento de padrão com tempo proporcional a M + N no pior caso.
- Knuth e Pratt, seguindo a construção que Cook usou na demonstração do seu teorema obtiveram um algoritmo relativamente simples e prático.
- Ocorreu também que Morris descobriu praticamente o mesmo algoritmo como solução de um problema de edição de texto.
- Os três cientistas (Knuth, Morris e Pratt) não se preocuparam em publicar o algoritmo até 1976.

Histórico

Histórico

- Nesse meio tempo, Boyer e Moore descobriram um algoritmo que é muito mais rápido em muitas aplicações.

Histórico

- Nesse meio tempo, Boyer e Moore descobriram um algoritmo que é muito mais rápido em muitas aplicações.
- Muitos editores de texto usam esse algoritmo para busca de cadeias.

Histórico

- Nesse meio tempo, Boyer e Moore descobriram um algoritmo que é muito mais rápido em muitas aplicações.
- Muitos editores de texto usam esse algoritmo para busca de cadeias.
- Em 1980, Rabin e Karp desenvolveram um algoritmo tão simples quanto o de força bruta que roda virtualmente sempre em tempo proporcional a M + N.

Histórico

- Nesse meio tempo, Boyer e Moore descobriram um algoritmo que é muito mais rápido em muitas aplicações.
- Muitos editores de texto usam esse algoritmo para busca de cadeias.
- Em 1980, Rabin e Karp desenvolveram um algoritmo tão simples quanto o de força bruta que roda virtualmente sempre em tempo proporcional a M + N.
- Além disso, o algoritmo deles estende-se facilmente a padrões bidimensionais que o torna mais útil que os outros para processamento de figuras.

FORÇA BRUTA

Força Bruta

Força Bruta

- O algoritmo de força bruta é o algoritmo mais simples para casamento de cadeias de caracteres.

Força Bruta

- O algoritmo de força bruta é o algoritmo mais simples para casamento de cadeias de caracteres.
- A ideia consiste em tentar casar qualquer subcadeia no texto de comprimento `m` com o padrão.

Força Bruta

- O algoritmo de força bruta é o algoritmo mais simples para casamento de cadeias de caracteres.
- A ideia consiste em tentar casar qualquer subcadeia no texto de comprimento `m` com o padrão.
- O pior caso do algoritmo de força bruta é:

Força Bruta

- O algoritmo de força bruta é o algoritmo mais simples para casamento de cadeias de caracteres.
- A ideia consiste em tentar casar qualquer subcadeia no texto de comprimento **m** com o padrão.
- O pior caso do algoritmo de força bruta é:
  - $C_n = m \times n$,

Força Bruta

- O algoritmo de força bruta é o algoritmo mais simples para casamento de cadeias de caracteres.
- A ideia consiste em tentar casar qualquer subcadeia no texto de comprimento **m** com o padrão.
- O pior caso do algoritmo de força bruta é:
  - $C_n = m \times n$,
- Isso ocorre, por exemplo, quando **P = aab** e **T = aaaaaaa**

Força Bruta

Força Bruta

- O algoritmo executa dois laços de repetição aninhados, sendo que o laço externo pesquisa todos os possíveis índices do padrão do texto, e o laço interno pesquisa cada caractere do padrão, comparando-o a seu correspondente potencial no texto.

Força Bruta

- O algoritmo executa dois laços de repetição aninhados, sendo que o laço externo pesquisa todos os possíveis índices do padrão do texto, e o laço interno pesquisa cada caractere do padrão, comparando-o a seu correspondente potencial no texto.
- O pior caso de tempo de procura não é razoável, pois para cada índice-candidato em **T** pode-se realizar até **m** comparações de caracteres para descobrir que **P** não é igual a **T** começando no índice atual.

Força Bruta

Força Bruta

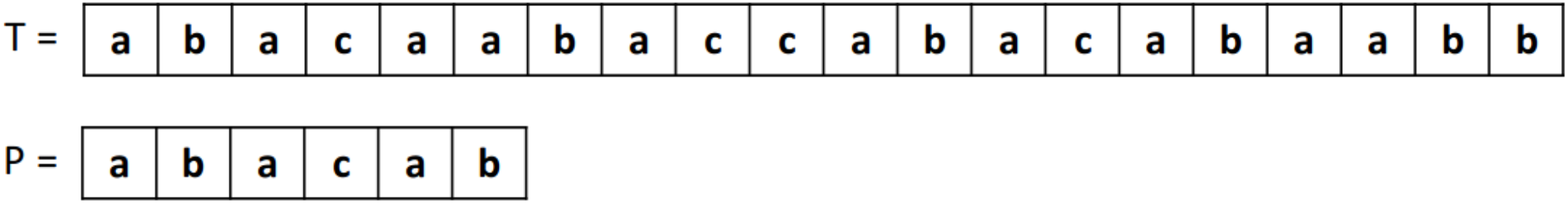
- Objetivo: descobrir se a string P (padrão) está em T.

Força Bruta

- Objetivo: descobrir se a string P (padrão) está em T.
- Força bruta: percorre elemento a elemento da esquerda para a direita.

Força Bruta

- Objetivo: descobrir se a string P (padrão) está em T.
- Força bruta: percorre elemento a elemento da esquerda para a direita.

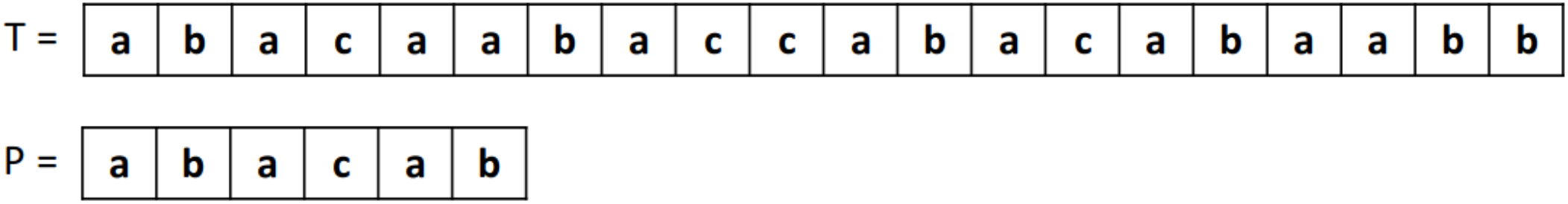


Força Bruta

- Algoritmo

Força Bruta

- Algoritmo



Força Bruta

- Algoritmo

T =

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

P =

a	b	a	c	a	b

```
1. busca_padrao(t, n, p, m)
2.   para i ← 0 até n - m faça
3.     j ← 0
4.     enquanto (j < m) e t[i + j] == p[j] faça
5.       j ← j + 1
6.     se j == m então
7.       retorne i
8.   retorne "padrão p não encontrado em t"
```

O(nm)

Força Bruta

Força Bruta

- O grande problema desse algoritmo é que nenhuma informação é guardada para ser usada em comparações futuras.

Força Bruta

- O grande problema desse algoritmo é que nenhuma informação é guardada para ser usada em comparações futuras.
- Características

Força Bruta

- O grande problema desse algoritmo é que nenhuma informação é guardada para ser usada em comparações futuras.
- Características
  - Não existe fase de pré-processamento.

Força Bruta

- O grande problema desse algoritmo é que nenhuma informação é guardada para ser usada em comparações futuras.
- Características
  - Não existe fase de pré-processamento.
  - Sempre desloca a janela de uma posição para a direita.

Força Bruta

- O grande problema desse algoritmo é que nenhuma informação é guardada para ser usada em comparações futuras.
- Características
  - Não existe fase de pré-processamento.
  - Sempre desloca a janela de uma posição para a direita.
  - Comparação dos caracteres pode ser feita em qualquer ordem.

Força Bruta

- O grande problema desse algoritmo é que nenhuma informação é guardada para ser usada em comparações futuras.
- Características
  - Não existe fase de pré-processamento.
  - Sempre desloca a janela de uma posição para a direita.
  - Comparação dos caracteres pode ser feita em qualquer ordem.
  - Complexidade é O(m.n).

Algoritmos Baseados em Autômatos

Algoritmos baseados em Autômatos

Algoritmos baseados em Autômatos

- Autômato é um modelo de computação (máquina) cujo propósito principal é reconhecer linguagens.

Algoritmos baseados em Autômatos

- Autômato é um modelo de computação (máquina) cujo propósito principal é reconhecer linguagens.
- Um autômato finito **M** é uma quíntupla

  $(Q, q_0, A, \Sigma, \delta)$ onde:

Algoritmos baseados em Autômatos

- Autômato é um modelo de computação (máquina) cujo propósito principal é reconhecer linguagens.
- Um autômato finito **M** é uma quíntupla

  $(Q, q_0, A, \Sigma, \delta)$ onde:

  - Q é um conjunto finito de **estados**.

Algoritmos baseados em Autômatos

- Autômato é um modelo de computação (máquina) cujo propósito principal é reconhecer linguagens.
- Um autômato finito **M** é uma quíntupla

  $(Q, q_0, A, \Sigma, \delta)$ onde:

  - Q é um conjunto finito de **estados**.
  - $q_0 \in Q$ é o **estado inicial**.

Algoritmos baseados em Autômatos

- Autômato é um modelo de computação (máquina) cujo propósito principal é reconhecer linguagens.
- Um autômato finito **M** é uma quíntupla

  $(Q, q_0, A, \Sigma, \delta)$ onde:

  - Q é um conjunto finito de **estados**.
  - $q_0 \in Q$ é o **estado inicial**.
  - $A \subseteq Q$ é o conjunto de **estados finais**.

Algoritmos baseados em Autômatos

- Autômato é um modelo de computação (máquina) cujo propósito principal é reconhecer linguagens.
- Um autômato finito **M** é uma quíntupla

  $(Q, q_0, A, \Sigma, \delta)$ onde:

  - Q é um conjunto finito de **estados**.
  - $q_0 \in Q$ é o **estado inicial**.
  - $A \subseteq Q$ é o conjunto de **estados finais**.
  - Σ é um **alfabeto** de entrada finito.

Algoritmos baseados em Autômatos

- Autômato é um modelo de computação (máquina) cujo propósito principal é reconhecer linguagens.
- Um autômato finito **M** é uma quíntupla

  $(Q, q_0, A, \Sigma, \delta)$ onde:

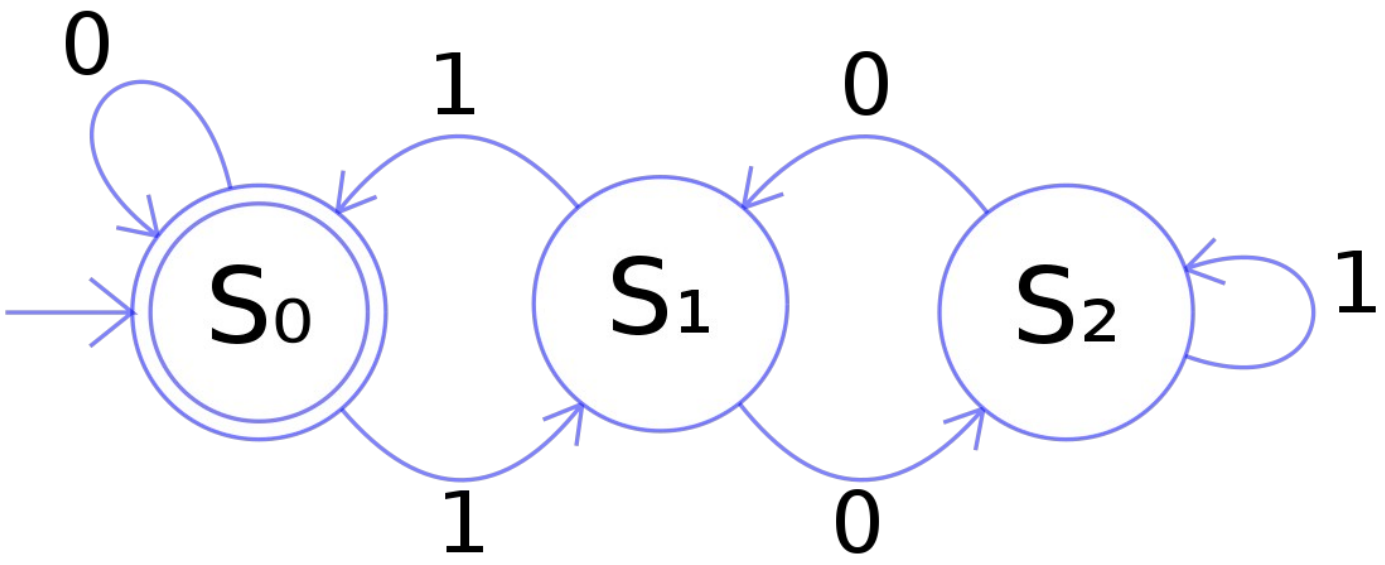
  - Q é um conjunto finito de **estados**.
  - $q_0 \in Q$ é o **estado inicial**.
  - $A \subseteq Q$ é o conjunto de **estados finais**.
  - Σ é um **alfabeto** de entrada finito.
  - δ é uma função de transição $Q \times \Sigma \rightarrow Q$.

Algoritmos baseados em Autômatos

- Exemplo

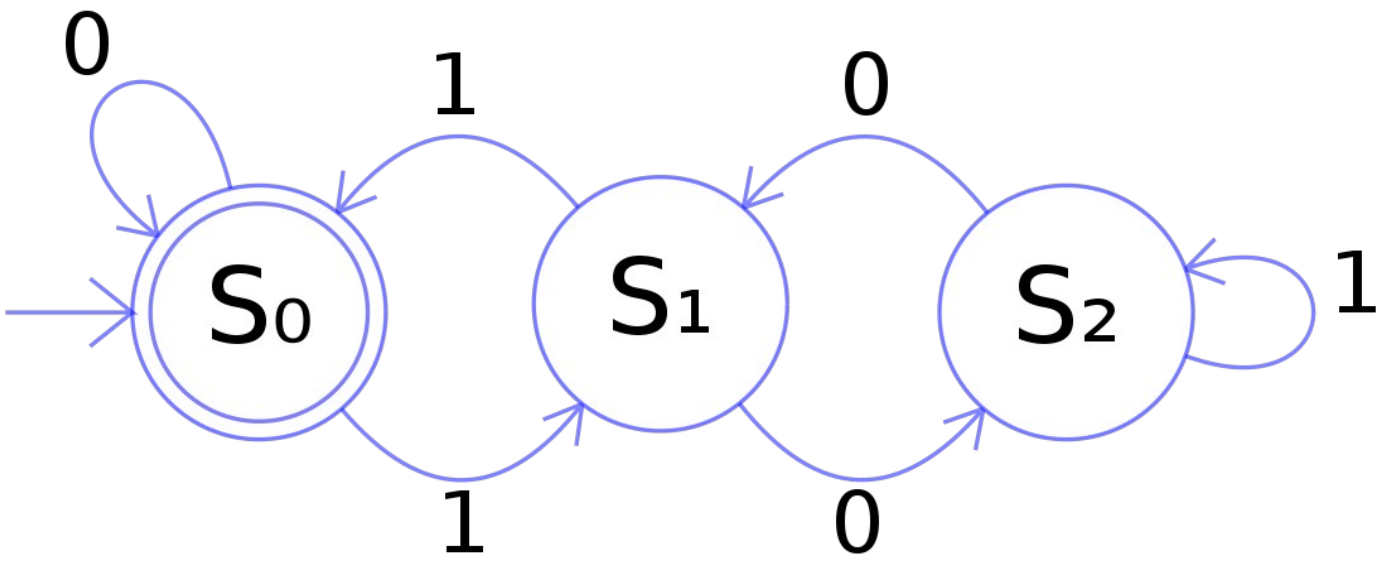
Algoritmos baseados em Autômatos

- Exemplo



Algoritmos baseados em Autômatos

- Exemplo



EM BREVE

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- É o primeiro algoritmo cujo pior caso tem complexidade de tempo linear no tamanho do texto.

Knuth-Morris-Pratt (KMP)

- É o primeiro algoritmo cujo pior caso tem complexidade de tempo linear no tamanho do texto.
- É um dos algoritmos mais famosos para resolver o problema de casamento de cadeias.

Knuth-Morris-Pratt (KMP)

- É o primeiro algoritmo cujo pior caso tem complexidade de tempo linear no tamanho do texto.
- É um dos algoritmos mais famosos para resolver o problema de casamento de cadeias.
- O algoritmo foi inventado por Donald Knuth e Vaughan Pratt e, independentemente, por James Morris em 1977.

Knuth-Morris-Pratt (KMP)

- É o primeiro algoritmo cujo pior caso tem complexidade de tempo linear no tamanho do texto.
- É um dos algoritmos mais famosos para resolver o problema de casamento de cadeias.
- O algoritmo foi inventado por Donald Knuth e Vaughan Pratt e, independentemente, por James Morris em 1977.
- Ele tem uma implementação complicada e, na prática, perde em eficiência para os algoritmos Shift-And e Boyer-Moore-Horspool.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Procura um padrão em um texto.

Knuth-Morris-Pratt (KMP)

- Procura um padrão em um texto.
- Pouco código e muito eficiente.

Knuth-Morris-Pratt (KMP)

- Procura um padrão em um texto.
- Pouco código e muito eficiente.
- Complexidade de O(n) no pior caso [linear].

Knuth-Morris-Pratt (KMP)

- Procura um padrão em um texto.
- Pouco código e muito eficiente.
- Complexidade de O(n) no pior caso [linear].
- A ideia geral do algoritmo:

Knuth-Morris-Pratt (KMP)

- Procura um padrão em um texto.
- Pouco código e muito eficiente.
- Complexidade de O(n) no pior caso [linear].
- A ideia geral do algoritmo:
  - Quando aparece uma diferença (conflito), a palavra tem, em si, a informação necessária para determinar aonde vai começar a próxima comparação.

Knuth-Morris-Pratt (KMP)

- Procura um padrão em um texto.
- Pouco código e muito eficiente.
- Complexidade de O(n) no pior caso [linear].
- A ideia geral do algoritmo:
  - Quando aparece uma diferença (conflito), a palavra tem, em si, a informação necessária para determinar aonde vai começar a próxima comparação.
  - Isso evita retroceder pelos caracteres já conhecidos.

Knuth-Morris-Pratt (KMP)

- Procura um padrão em um texto.
- Pouco código e muito eficiente.
- Complexidade de O(n) no pior caso [linear].
- A ideia geral do algoritmo:
  - Quando aparece uma diferença (conflito), a palavra tem, em si, a informação necessária para determinar aonde vai começar a próxima comparação.
  - Isso evita retroceder pelos caracteres já conhecidos.
  - Se for possível detectar uma falha (após algumas igualdades), nós já saberemos alguns dos caracteres no texto da próxima janela. Nós nos aproveitamos dessa informação para evitar a comparação de caracteres que nós já sabemos que serão iguais.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Exemplos (ruins) usando estratégia de força bruta:

Knuth-Morris-Pratt (KMP)

- Exemplos (ruins) usando estratégia de força bruta:
  - T = “AAAAAAAAAAAAAAAAAAB” e P = “AAAAB”.

Knuth-Morris-Pratt (KMP)

- Exemplos (ruins) usando estratégia de força bruta:
  - T = “AAAAAAAAAAAAAAAAAAB” e P = “AAAAB”.
  - T = “ABABABCABABABCABABABC” e

    P = “ABABAC”.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Exemplo: T = “AAAAABAAABA” e P = “AAAA”

Knuth-Morris-Pratt (KMP)

- Exemplo: T = “AAAAABAAABA” e P = “AAAA”
  - Nós comparamos a primeira janela de T com P: T = “**AAAA**ABAAABA” e P = “**AAAA**”.

Knuth-Morris-Pratt (KMP)

- Exemplo: T = “AAAAABAAABA” e P = “AAAA”
  - Nós comparamos a primeira janela de T com P: T = “**AAAA**ABAAABA” e P = “**AAAA**”.
  - No próximo passo, nós comparamos a próxima janela de T com P: T = “A**AAAA**BAAABA” e P = “**AAAA**”.

Knuth-Morris-Pratt (KMP)

- Exemplo: T = “AAAAABAAABA” e P = “AAAA”
  - Nós comparamos a primeira janela de T com P: T = “**AAAA**ABAAABA” e P = “**AAAA**”.
  - No próximo passo, nós comparamos a próxima janela de T com P: T = “A**AAAA**BAAABA” e P = “**AAAA**”.
  - No caso do KMP, nós só comparamos o próximo caractere de T com o quarto caractere de P, pois nós já sabemos que o três primeiros caracteres serão idênticos.

Knuth-Morris-Pratt (KMP)

- Exemplo: T = “AAAAABAAABA” e P = “AAAA”
  - Nós comparamos a primeira janela de T com P: T = “**AAAA**ABAAABA” e P = “**AAAA**”.
  - No próximo passo, nós comparamos a próxima janela de T com P: T = “A**AAAA**BAAABA” e P = “**AAAA**”.
  - No caso do KMP, nós só comparamos o próximo caractere de T com o quarto caractere de P, pois nós já sabemos que o três primeiros caracteres serão idênticos.
  - Para isso, é necessário uma função de pré-processamento.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Etapas:

Knuth-Morris-Pratt (KMP)

- Etapas:
  - Função prefixo.

Knuth-Morris-Pratt (KMP)

- Etapas:
  - Função prefixo.
  - Função de comparação.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Função prefixo

Knuth-Morris-Pratt (KMP)

- Função prefixo
  - É feito um pré-processamento do padrão e construído um vetor auxiliar de tamanho **m**.

Knuth-Morris-Pratt (KMP)

- Função prefixo
  - É feito um pré-processamento do padrão e construído um vetor auxiliar de tamanho **m**.
  - O pré-processamento analisa todos os prefixos do padrão procurando pelo maior sufixo destes prefixos que também seja prefixo.

Knuth-Morris-Pratt (KMP)

- Função prefixo
  - É feito um pré-processamento do padrão e construído um vetor auxiliar de tamanho **m**.
  - O pré-processamento analisa todos os prefixos do padrão procurando pelo maior sufixo destes prefixos que também seja prefixo.
  - Desta maneira, evita que um caractere seja reexaminado.

Knuth-Morris-Pratt (KMP)

- Função prefixo
  - É feito um pré-processamento do padrão e construído um vetor auxiliar de tamanho **m**.
  - O pré-processamento analisa todos os prefixos do padrão procurando pelo maior sufixo destes prefixos que também seja prefixo.
  - Desta maneira, evita que um caractere seja reexaminado.
  - Ao final, teremos um vetor que conterá cada posição que gera um prefixo para cada posição do padrão.

Knuth-Morris-Pratt (KMP)

- Função prefixo
  - É feito um pré-processamento do padrão e construído um vetor auxiliar de tamanho **m**.
  - O pré-processamento analisa todos os prefixos do padrão procurando pelo maior sufixo destes prefixos que também seja prefixo.
  - Desta maneira, evita que um caractere seja reexaminado.
  - Ao final, teremos um vetor que conterá cada posição que gera um prefixo para cada posição do padrão.
  - O pré-processamento é realizado no padrão para determinar se os seus prefixos aparecem como subsequências deles mesmos.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”
  - F = [0, 1, 2, 3]

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”
  - F = [0, 1, 2, 3]
  - P = “ABCDE”

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”
  - F = [0, 1, 2, 3]
  - P = “ABCDE”
  - F = [0, 0, 0, 0, 0]

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”
  - F = [0, 1, 2, 3]
  - P = “ABCDE”
  - F = [0, 0, 0, 0, 0]
  - P = “AABAACAABAA”

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”
  - F = [0, 1, 2, 3]
  - P = “ABCDE”
  - F = [0, 0, 0, 0, 0]
  - P = “AABAACAABAA”
  - F = [0, 1, 0, 1, 2, 0, 1, 2, 3, 4, 5]

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”
  - F = [0, 1, 2, 3]
  - P = “ABCDE”
  - F = [0, 0, 0, 0, 0]
  - P = “AABAACAABAA”
  - F = [0, 1, 0, 1, 2, 0, 1, 2, 3, 4, 5]
  - P = “AAACAAAAAC”

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”
  - F = [0, 1, 2, 3]
  - P = “ABCDE”
  - F = [0, 0, 0, 0, 0]
  - P = “AABAACAABAA”
  - F = [0, 1, 0, 1, 2, 0, 1, 2, 3, 4, 5]
  - P = “AAACAAAAAC”
  - F = [0, 1, 2, 0, 1, 2, 3, 3, 3, 4]

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”
  - F = [0, 1, 2, 3]
  - P = “ABCDE”
  - F = [0, 0, 0, 0, 0]
  - P = “AABAACAABAA”
  - F = [0, 1, 0, 1, 2, 0, 1, 2, 3, 4, 5]
  - P = “AAACAAAAAC”
  - F = [0, 1, 2, 0, 1, 2, 3, 3, 3, 4]
  - P = “AAABAAA”

Knuth-Morris-Pratt (KMP)

- Exemplos de função prefixo:
  - P = “AAAA”
  - F = [0, 1, 2, 3]
  - P = “ABCDE”
  - F = [0, 0, 0, 0, 0]
  - P = “AABAACAABAA”
  - F = [0, 1, 0, 1, 2, 0, 1, 2, 3, 4, 5]
  - P = “AAACAAAAAC”
  - F = [0, 1, 2, 0, 1, 2, 3, 3, 3, 4]
  - P = “AAABAAA”
  - F = [0, 1, 2, 0, 1, 2, 3]

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Algoritmo de pré-processamento:

Knuth-Morris-Pratt (KMP)

- Algoritmo de pré-processamento:
  - Nós calculamos os valores F[ ]. Para isto, nós acompanhamos o valor do sufixo do prefixo mais longo para o índice anterior.

Knuth-Morris-Pratt (KMP)

- Algoritmo de pré-processamento:
  - Nós calculamos os valores F[ ]. Para isto, nós acompanhamos o valor do sufixo do prefixo mais longo para o índice anterior.
  - Nós inicializamos F[0] e tamanho como 0 (zero).

Knuth-Morris-Pratt (KMP)

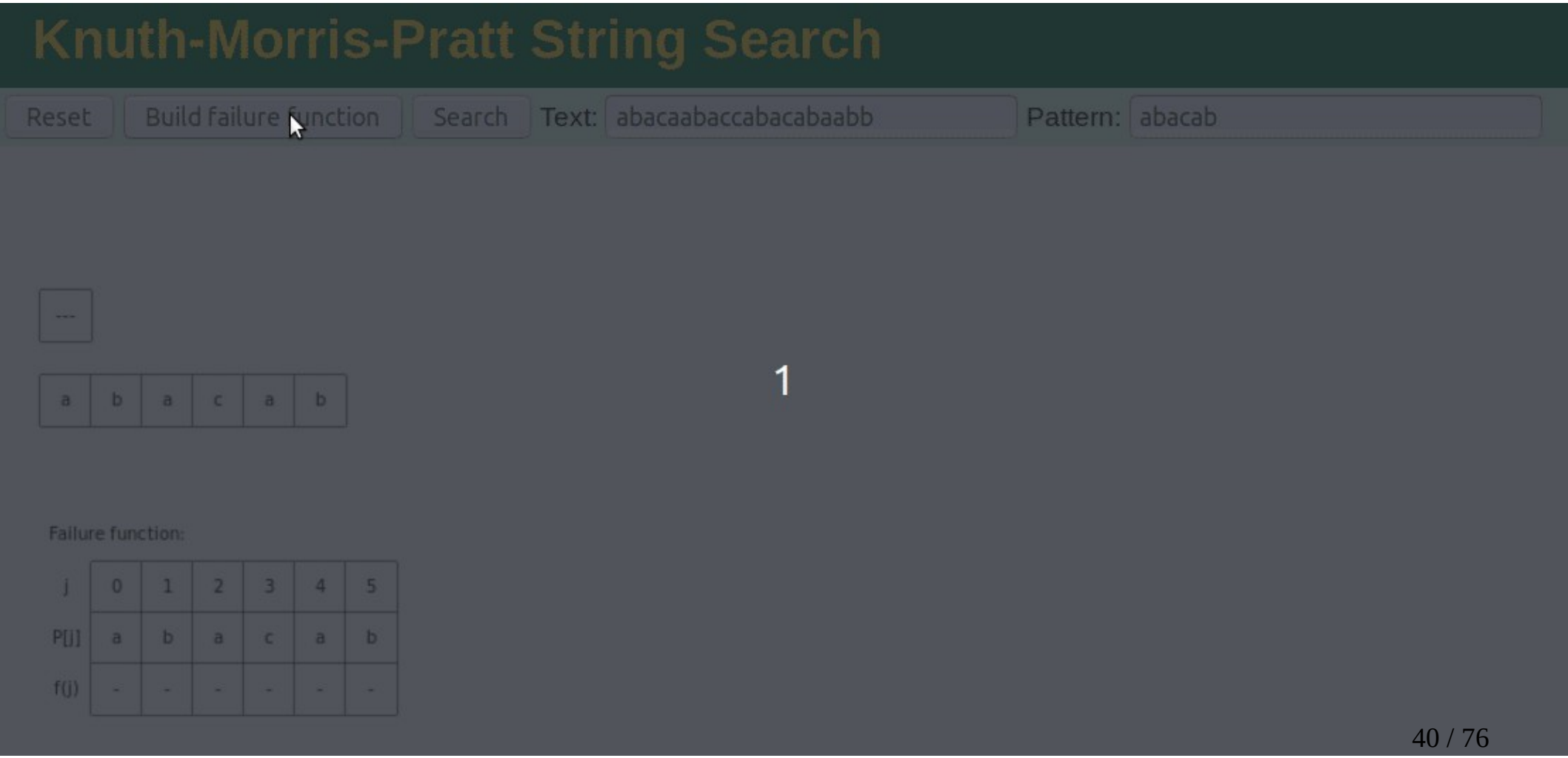
- Algoritmo de pré-processamento:
  - Nós calculamos os valores F[ ]. Para isto, nós acompanhamos o valor do sufixo do prefixo mais longo para o índice anterior.
  - Nós inicializamos F[0] e tamanho como 0 (zero).
  - Se o padrao[tamanho] e o padrao[i] forem iguais, nós incrementamos o tamanho de 1 e atribuímos o valor incrementado para F.

Knuth-Morris-Pratt (KMP)

- Algoritmo de pré-processamento:
  - Nós calculamos os valores F[ ]. Para isto, nós acompanhamos o valor do sufixo do prefixo mais longo para o índice anterior.
  - Nós inicializamos F[0] e tamanho como 0 (zero).
  - Se o padrao[tamanho] e o padrao[i] forem iguais, nós incrementamos o tamanho de 1 e atribuímos o valor incrementado para F.
  - Se o padrao[tamanho] e o padrao[i] não forem iguais e o tamanho não for 0 (zero), então nós atualizamos tamanho para F[tamanho-1].

Knuth-Morris-Pratt (KMP)

- Função prefixo



Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Na função de comparação, diferentemente dos algoritmos de força-bruta, nós usamos os valores da função F (pré-processamento) para decidir os próximos caracteres que serão comparados.

Knuth-Morris-Pratt (KMP)

- Na função de comparação, diferentemente dos algoritmos de força-bruta, nós usamos os valores da função F (pré-processamento) para decidir os próximos caracteres que serão comparados.
- O objetivo é **não comparar caracteres** que **nós já sabemos** que são iguais.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Algoritmo de comparação:

Knuth-Morris-Pratt (KMP)

- Algoritmo de comparação:
  - Nós iniciamos a comparação do P[j] com j = 0 com os caracteres da atual janela do texto.

Knuth-Morris-Pratt (KMP)

- Algoritmo de comparação:
  - Nós iniciamos a comparação do P[j] com j = 0 com os caracteres da atual janela do texto.
  - Nós avançamos na comparação de T[i] e P[j] e continuamos a incrementar i e j enquanto P[j] e T[i] forem iguais.

Knuth-Morris-Pratt (KMP)

- Algoritmo de comparação:
  - Nós iniciamos a comparação do P[j] com j = 0 com os caracteres da atual janela do texto.
  - Nós avançamos na comparação de T[i] e P[j] e continuamos a incrementar i e j enquanto P[j] e T[i] forem iguais.
  - Quando ocorrer uma desigualdade (falha):

Knuth-Morris-Pratt (KMP)

- Algoritmo de comparação:
  - Nós iniciamos a comparação do P[j] com j = 0 com os caracteres da atual janela do texto.
  - Nós avançamos na comparação de T[i] e P[j] e continuamos a incrementar i e j enquanto P[j] e T[i] forem iguais.
  - Quando ocorrer uma desigualdade (falha):
    - Nós sabemos que os caracteres P[0..j-1] são iguais com T[i-j...i-1].

Knuth-Morris-Pratt (KMP)

- Algoritmo de comparação:
  - Nós iniciamos a comparação do P[j] com j = 0 com os caracteres da atual janela do texto.
  - Nós avançamos na comparação de T[i] e P[j] e continuamos a incrementar i e j enquanto P[j] e T[i] forem iguais.
  - Quando ocorrer uma desigualdade (falha):
    - Nós sabemos que os caracteres P[0..j-1] são iguais com T[i-j...i-1].
    - Nós também sabemos que F[j-1] é a quantidade de caracteres de P[0..j-1] que são tanto prefixo como sufixo.

Knuth-Morris-Pratt (KMP)

- Algoritmo de comparação:
  - Nós iniciamos a comparação do P[j] com j = 0 com os caracteres da atual janela do texto.
  - Nós avançamos na comparação de T[i] e P[j] e continuamos a incrementar i e j enquanto P[j] e T[i] forem iguais.
  - Quando ocorrer uma desigualdade (falha):
    - Nós sabemos que os caracteres P[0..j-1] são iguais com T[i-j...i-1].
    - Nós também sabemos que F[j-1] é a quantidade de caracteres de P[0..j-1] que são tanto prefixo como sufixo.
    - A partir dessa análise, nós podemos concluir que não precisamos comparar os caracteres F[j-1] com T[i-j..i-1] porque nós sabemos que esses caracteres serão iguais de qualquer forma.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

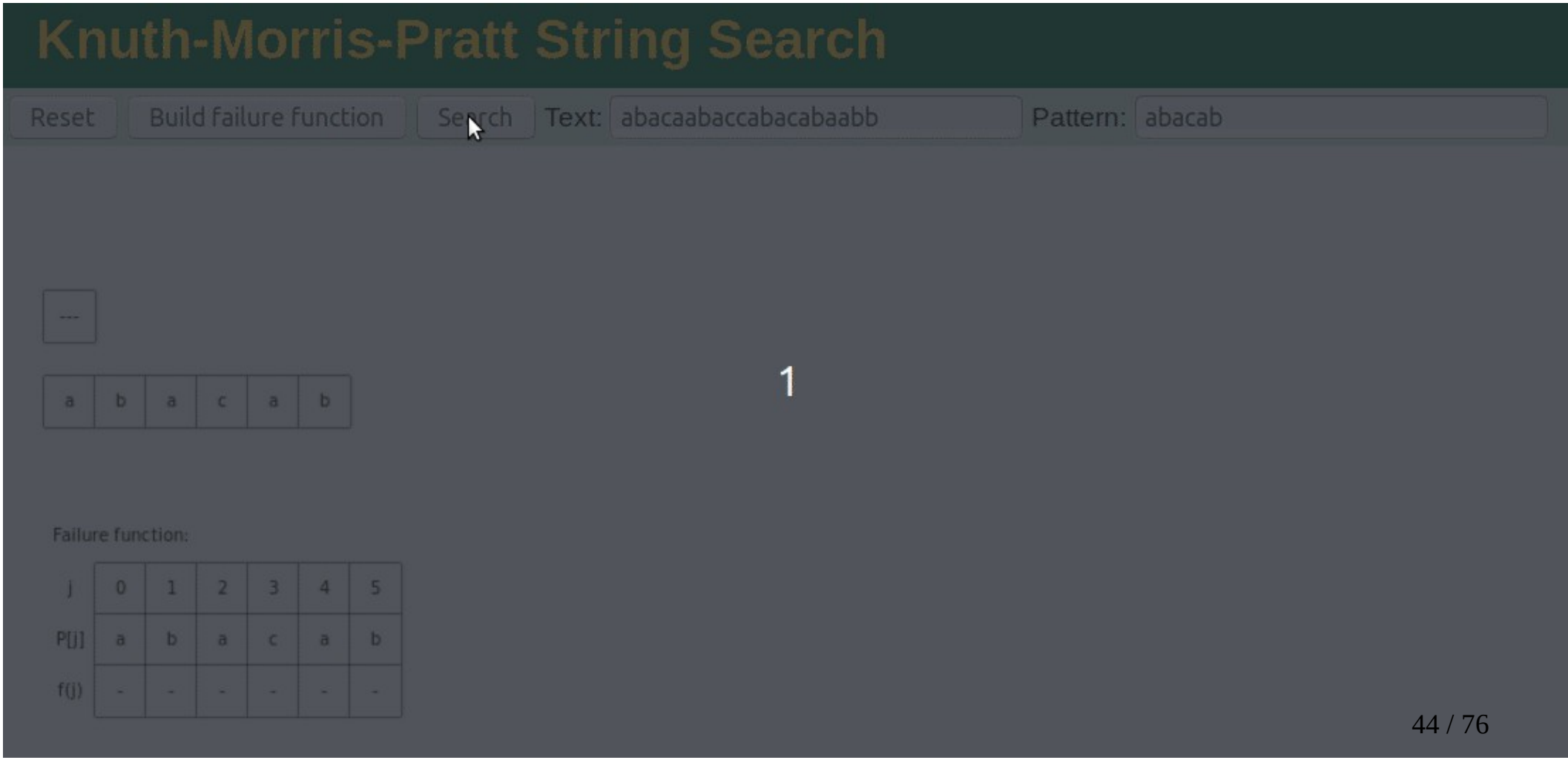
- Exemplo de comparação:

Knuth-Morris-Pratt (KMP)

- Exemplo de comparação:
  - T = “AAAAABAAABA” , P = “AAAA”
  - F = {0, 1, 2, 3}

Knuth-Morris-Pratt (KMP)

- Exemplo



Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Prefixo:

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Prefixo:
  - **Passo 1.** Crie um *array* com tamanho igual ao padrão: LPS[tam].

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Prefixo:
  - **Passo 1.** Crie um *array* com tamanho igual ao padrão: LPS[tam].
  - **Passo 2.** Crie as variáveis i e j, onde i = 0, j = 1 e LPS[0] = 0.

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Prefixo:
  - **Passo 1.** Crie um *array* com tamanho igual ao padrão: LPS[tam].
  - **Passo 2.** Crie as variáveis i e j, onde i = 0, j = 1 e LPS[0] = 0.
  - **Passo 3.** Compare os itens de Padrao[i] e Padrao[j].

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Prefixo:
  - **Passo 1.** Crie um *array* com tamanho igual ao padrão: LPS[tam].
  - **Passo 2.** Crie as variáveis i e j, onde i = 0, j = 1 e LPS[0] = 0.
  - **Passo 3.** Compare os itens de Padrao[i] e Padrao[j].
  - **Passo 4.** Se forem iguais, então LPS[j] = i + 1, incrementa i e j, retorna Passo 3.

Knuth-Morris-Pratt (KMP)

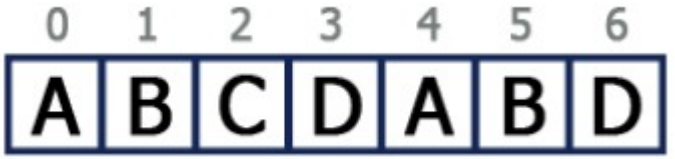
- Algoritmo Função Prefixo:
  - **Passo 1.** Crie um *array* com tamanho igual ao padrão: LPS[tam].
  - **Passo 2.** Crie as variáveis i e j, onde i = 0, j = 1 e LPS[0] = 0.
  - **Passo 3.** Compare os itens de Padrao[i] e Padrao[j].
  - **Passo 4.** Se forem iguais, então LPS[j] = i + 1, incrementa i e j, retorna Passo 3.
  - **Passo 5.** Se não forem iguais, então verifique o valor de i. Se i = 0, então LPS[j] = 0 e incrementa j. Se i ≠ 0, então i = LPS[i-1] e retorna Passo 3.

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Prefixo:
  - **Passo 1.** Crie um *array* com tamanho igual ao padrão: LPS[tam].
  - **Passo 2.** Crie as variáveis i e j, onde i = 0, j = 1 e LPS[0] = 0.
  - **Passo 3.** Compare os itens de Padrao[i] e Padrao[j].
  - **Passo 4.** Se forem iguais, então LPS[j] = i + 1, incrementa i e j, retorna Passo 3.
  - **Passo 5.** Se não forem iguais, então verifique o valor de i. Se i = 0, então LPS[j] = 0 e incrementa j. Se i ≠ 0, então i = LPS[i-1] e retorna Passo 3.
  - **Passo 6.** Repita os passos anteriores até que todos os valores de LPS sejam preenchidos.

Knuth-Morris-Pratt (KMP)

- Calcule a função de prefixo do padrão abaixo:



Knuth-Morris-Pratt (KMP)

- Função de comparação:
  - Nós utilizamos o *array* LPS para decidir quantos caracteres (posições) serão saltadas para comparação quando uma falha ocorrer.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Comparação:

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Comparação:
  - Se houver uma falha, verifique o valor de LPS do caracter anterior do caracter da falha no padrão.

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Comparação:
  - Se houver uma falha, verifique o valor de LPS do caracter anterior do caracter da falha no padrão.
  - Se esse valor for igual a 0 (zero), então comece a comparar o primeiro caracter do padrão com o próximo caracter da falha no texto.

Knuth-Morris-Pratt (KMP)

- Algoritmo Função Comparação:
  - Se houver uma falha, verifique o valor de LPS do caracter anterior do caracter da falha no padrão.
  - Se esse valor for igual a 0 (zero), então comece a comparar o primeiro caracter do padrão com o próximo caracter da falha no texto.
  - Se esse valor não for igual a 0 (zero), então comece a comparar o caracter que está em um valor de índice igual ao valor LPS do caracter anterior com o caracter que houve a falha no padrão com o caracter da falha no texto.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Exemplo[1]:

1. http://www.btechsmartclass.com/data_structures/knuth-morris-pratt-algorithm.html

Knuth-Morris-Pratt (KMP)

- Exemplo[1]:

Text : ABC ABCDAB ABCDABCDABDE

Pattern : ABCDABD

1. http://www.btechsmartclass.com/data_structures/knuth-morris-pratt-algorithm.html

Knuth-Morris-Pratt (KMP)

- Exemplo[1]:

Text : ABC ABCDAB ABCDABCDABDE
Pattern : ABCDABD

- Função de Prefixo:

1. http://www.btechsmartclass.com/data_structures/knuth-morris-pratt-algorithm.html

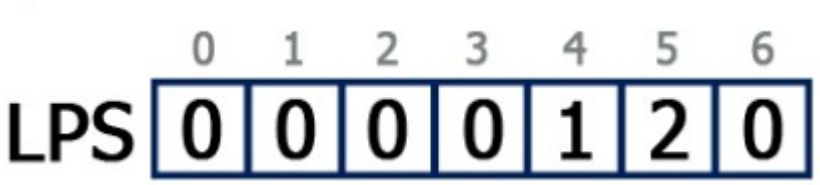
Knuth-Morris-Pratt (KMP)

- Exemplo[1]:

Text : ABC ABCDAB ABCDABCDABDE

Pattern : ABCDABD

- Função de Prefixo:



1. http://www.btechsmartclass.com/data_structures/knuth-morris-pratt-algorithm.html

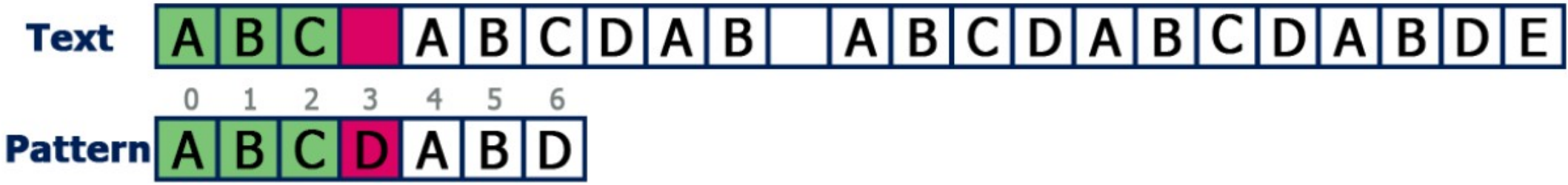
Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Passo 1

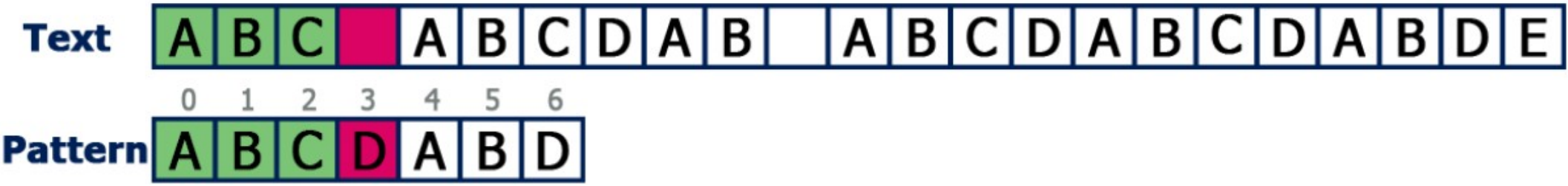
Knuth-Morris-Pratt (KMP)

- Passo 1



Knuth-Morris-Pratt (KMP)

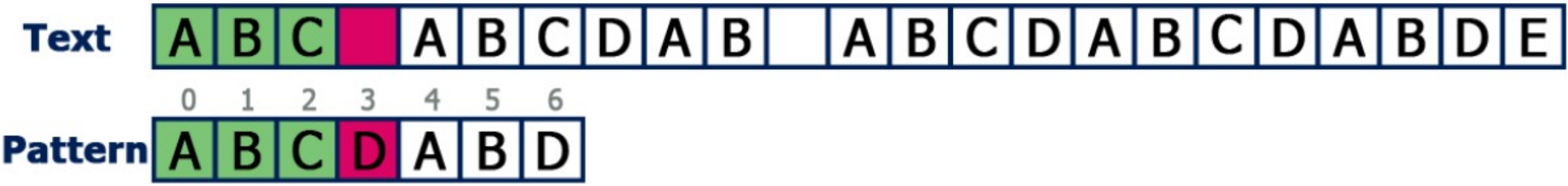
- Passo 1



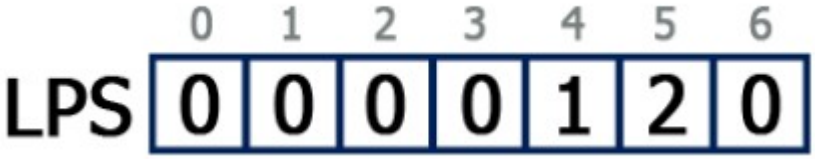
  - Houve a falha no Pattern[3], então devemos considerar o valor LPS[2].

Knuth-Morris-Pratt (KMP)

- Passo 1

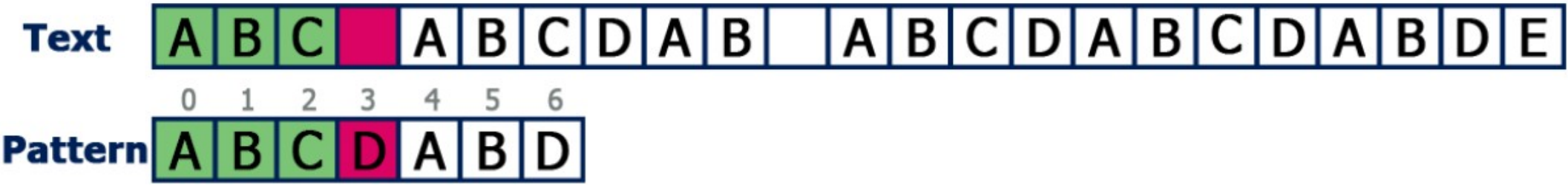


  - Houve a falha no Pattern[3], então devemos considerar o valor LPS[2].

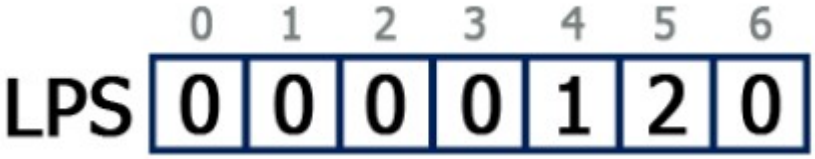


Knuth-Morris-Pratt (KMP)

- Passo 1



  - Houve a falha no Pattern[3], então devemos considerar o valor LPS[2].



  - Como LPS[2] = 0, então nós devemos comparar o primeiro caracter do Padrão (*Pattern*) com o próximo caracter do Texto (*Text*).

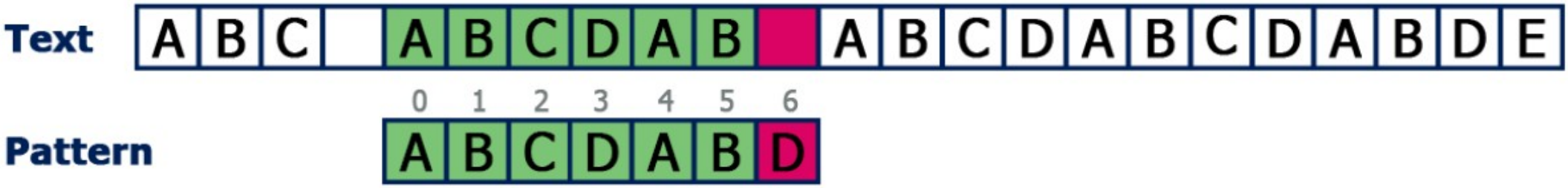
Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Passo 2

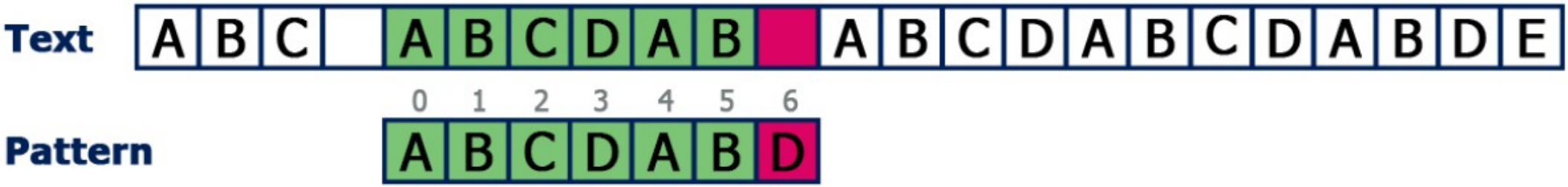
Knuth-Morris-Pratt (KMP)

- Passo 2



Knuth-Morris-Pratt (KMP)

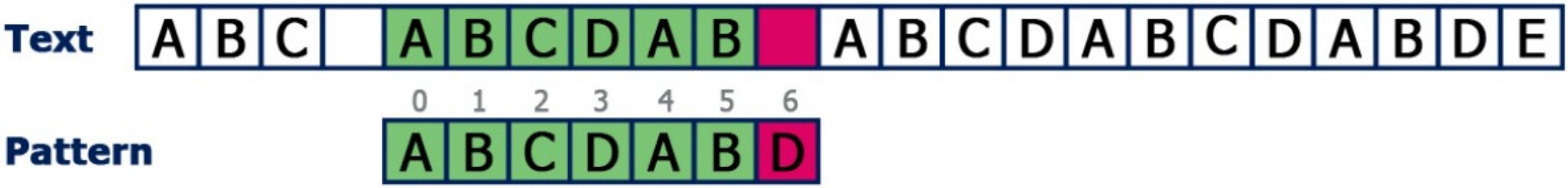
- Passo 2



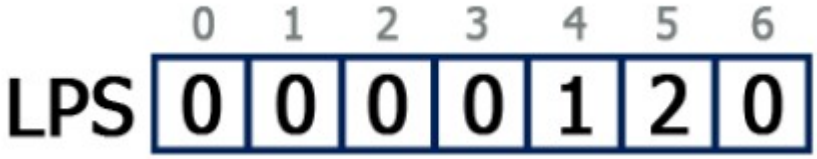
  - Houve uma falha no Pattern[6], então devemos considerar o valor LPS[5].

Knuth-Morris-Pratt (KMP)

- Passo 2

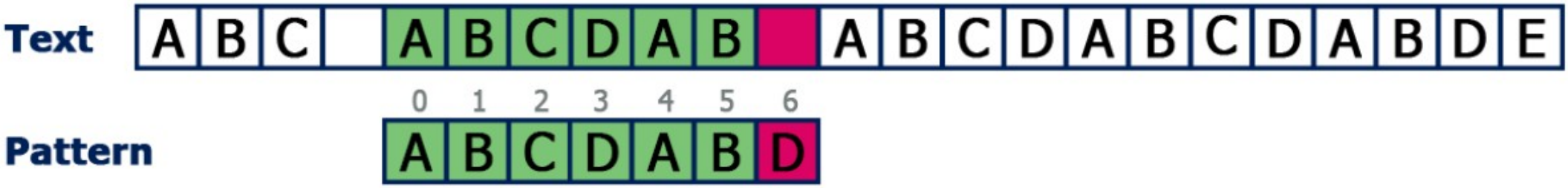


  - Houve uma falha no Pattern[6], então devemos considerar o valor LPS[5].

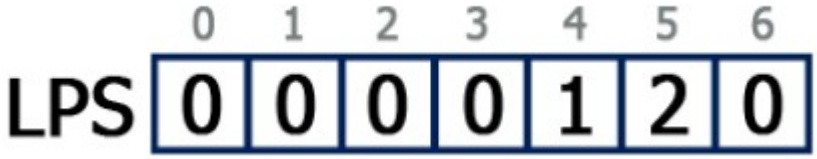


Knuth-Morris-Pratt (KMP)

- Passo 2



  - Houve uma falha no Pattern[6], então devemos considerar o valor LPS[5].



  - Como LPS[5] = 2, então nós comparamos Pattern[2] com o caracter que houve a falha.

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

- Passo 3

Knuth-Morris-Pratt (KMP)

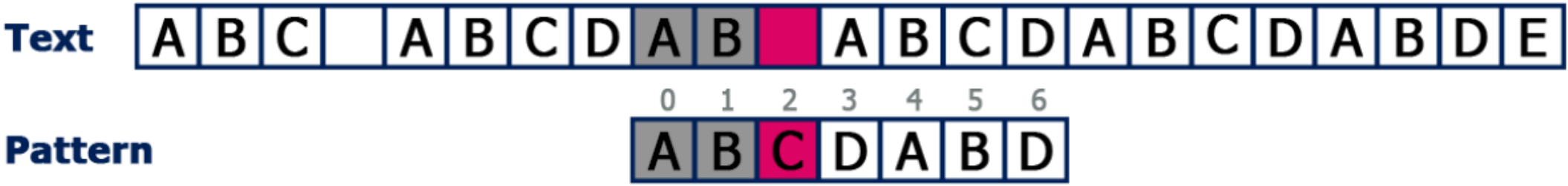
- Passo 3
  - Como o valor de LPS é 2, então não precisamos comparar com o Pattern[0] e Pattern[1].

Knuth-Morris-Pratt (KMP)

- Passo 3
  - Como o valor de LPS é 2, então não precisamos comparar com o Pattern[0] e Pattern[1].

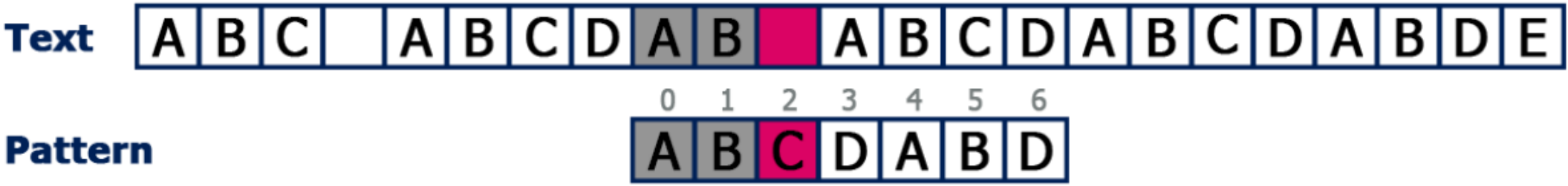
Knuth-Morris-Pratt (KMP)

- Passo 3
  - Como o valor de LPS é 2, então não precisamos comparar com o Pattern[0] e Pattern[1].



Knuth-Morris-Pratt (KMP)

- Passo 3
  - Como o valor de LPS é 2, então não precisamos comparar com o Pattern[0] e Pattern[1].

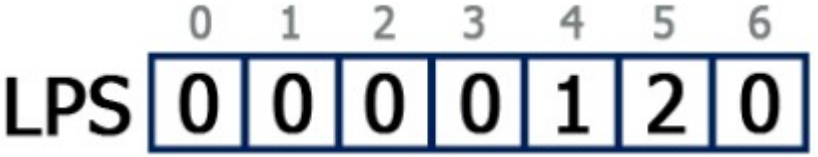


  - Houve uma falha no Pattern[2], então devemos considerar o valor LPS[1].

Knuth-Morris-Pratt (KMP)

- Passo 3
  - Como o valor de LPS é 2, então não precisamos comparar com o Pattern[0] e Pattern[1].

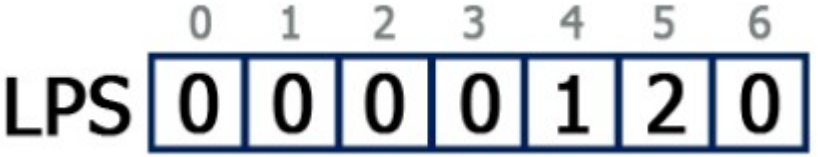
  - Houve uma falha no Pattern[2], então devemos considerar o valor LPS[1].



Knuth-Morris-Pratt (KMP)

- Passo 3
  - Como o valor de LPS é 2, então não precisamos comparar com o Pattern[0] e Pattern[1].

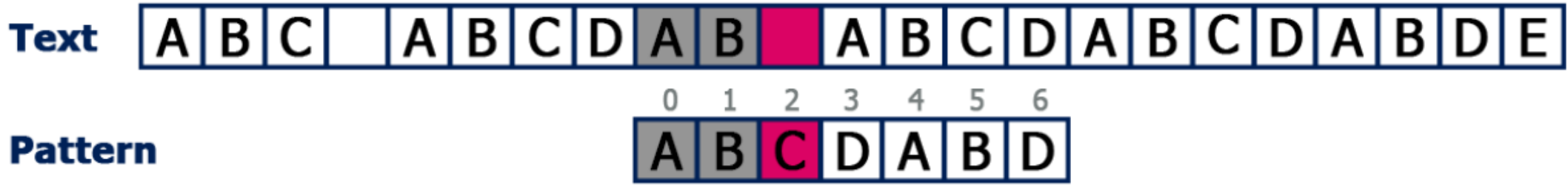
  - Houve uma falha no Pattern[2], então devemos considerar o valor LPS[1].



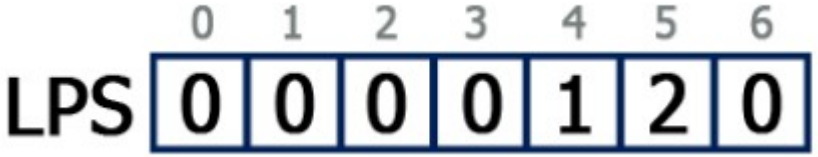
  - Como LPS[1] = 0, então nós devemos comparar o primeiro caracter do Padrão (*Pattern*) com o próximo caracter do Texto (*Text*).

Knuth-Morris-Pratt (KMP)

- Passo 3
  - Como o valor de LPS é 2, então não precisamos comparar com o Pattern[0] e Pattern[1].



  - Houve uma falha no Pattern[2], então devemos considerar o valor LPS[1].



  - Como LPS[1] = 0, então nós devemos comparar o primeiro caracter do Padrão (*Pattern*) com o próximo caracter do Texto (*Text*).

Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

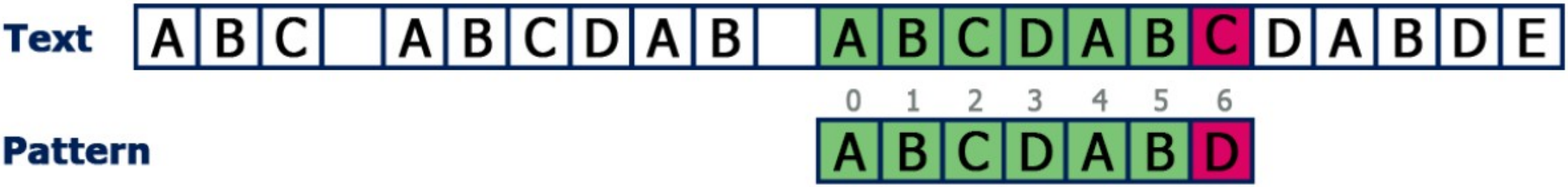
- Passo 4

Knuth-Morris-Pratt (KMP)

- Passo 4
  - Compare o Pattern[0] com o próximo caracter do texto.

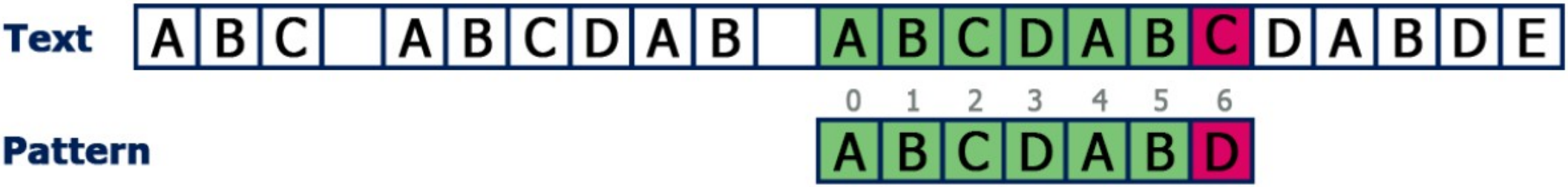
Knuth-Morris-Pratt (KMP)

- Passo 4
  - Compare o Pattern[0] com o próximo caracter do texto.



Knuth-Morris-Pratt (KMP)

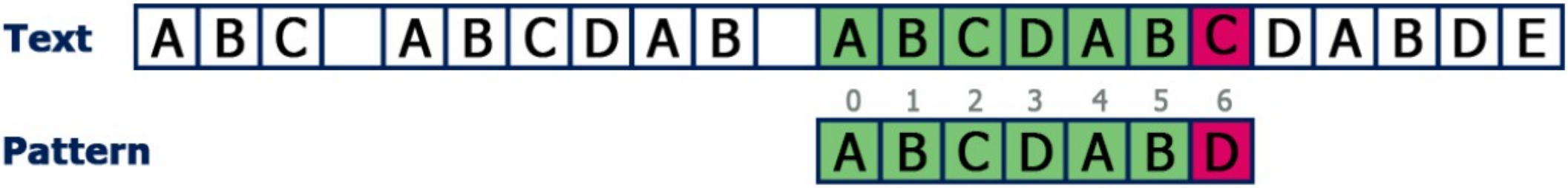
- Passo 4
  - Compare o Pattern[0] com o próximo caracter do texto.



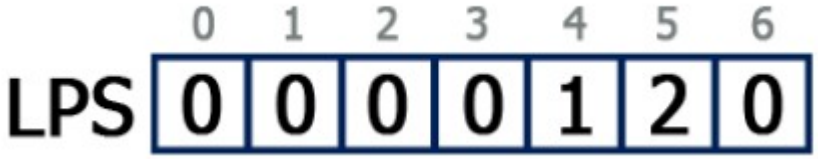
  - Houve uma falha no Pattern[6], então devemos considerar o valor LPS[5].

Knuth-Morris-Pratt (KMP)

- Passo 4
  - Compare o Pattern[0] com o próximo caracter do texto.

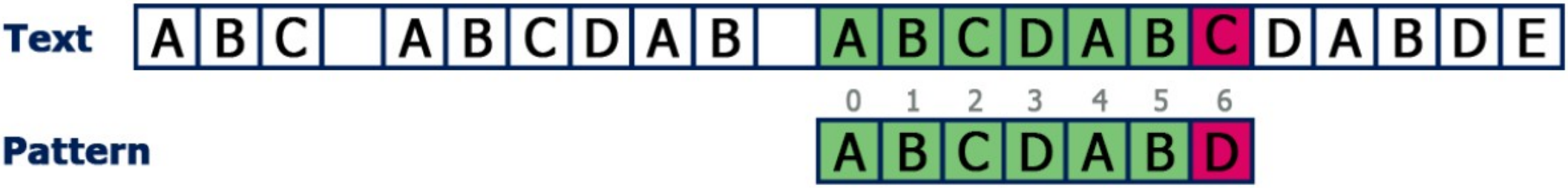


  - Houve uma falha no Pattern[6], então devemos considerar o valor LPS[5].

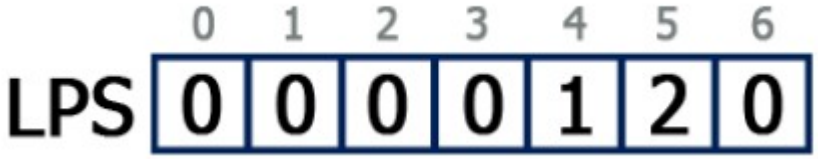


Knuth-Morris-Pratt (KMP)

- Passo 4
  - Compare o Pattern[0] com o próximo caracter do texto.

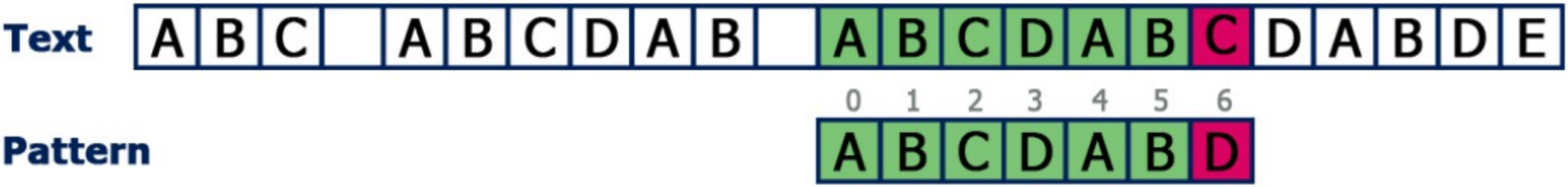


  - Houve uma falha no Pattern[6], então devemos considerar o valor LPS[5].
  - Como LPS[5] = 2, então nós comparamos Pattern[2] com o caracter que houve a falha.

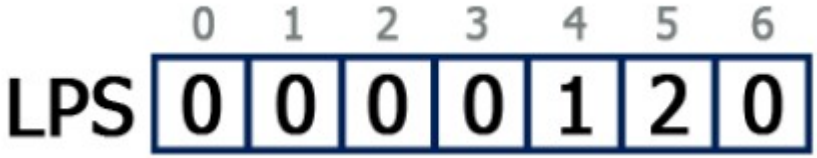


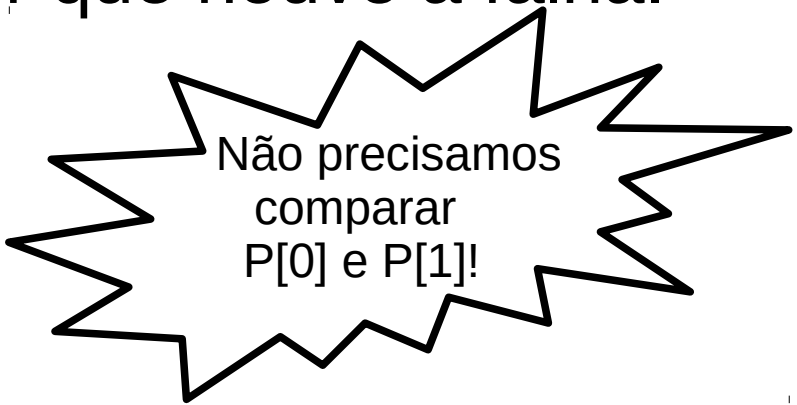
Knuth-Morris-Pratt (KMP)

- Passo 4
  - Compare o Pattern[0] com o próximo caracter do texto.



  - Houve uma falha no Pattern[6], então devemos considerar o valor LPS[5].
  - Como LPS[5] = 2, então nós comparamos Pattern[2] com o caracter que houve a falha.





Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt (KMP)

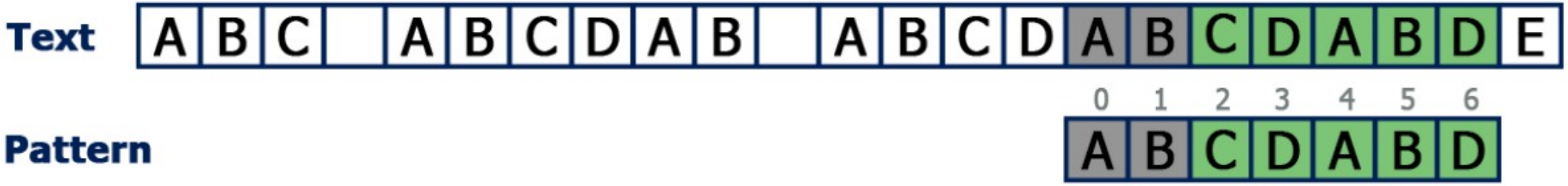
- Passo 5

Knuth-Morris-Pratt (KMP)

- Passo 5
  - Compare o Pattern[2] com o caracter que houve a falha no texto.

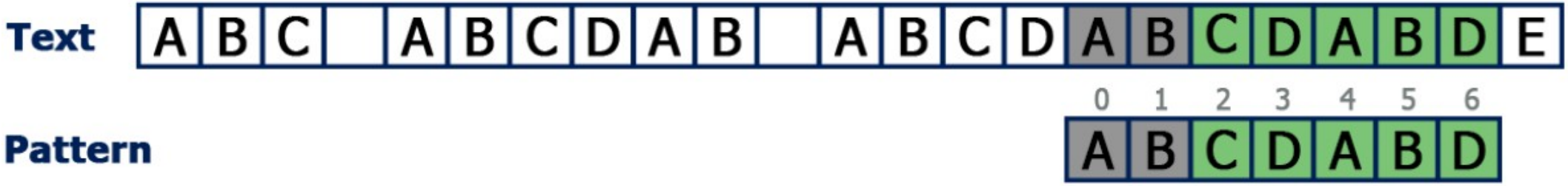
Knuth-Morris-Pratt (KMP)

- Passo 5
  - Compare o Pattern[2] com o caracter que houve a falha no texto.



Knuth-Morris-Pratt (KMP)

- Passo 5
  - Compare o Pattern[2] com o caracter que houve a falha no texto.



  - Agora todos os caracteres do Padrão (*Pattern*) são iguais com a substring do Texto (*Text*).

Algoritmo Boyer-Moore (BM)

Algoritmo BM

Algoritmo BM

- O algoritmo é um dos mais eficientes algoritmos de busca em strings.

Algoritmo BM

- O algoritmo é um dos mais eficientes algoritmos de busca em strings.
- Ele foi desenvolvido em 1977 pelos professores Robert Stephen **Boyer** e J Strother **Moore**.

Algoritmo BM

- O algoritmo é um dos mais eficientes algoritmos de busca em strings.
- Ele foi desenvolvido em 1977 pelos professores Robert Stephen **Boyer** e J Strother **Moore**.
- Também chamado de Boyer-Moore.

Algoritmo BM

- O algoritmo é um dos mais eficientes algoritmos de busca em strings.
- Ele foi desenvolvido em 1977 pelos professores Robert Stephen **Boyer** e J Strother **Moore**.
- Também chamado de Boyer-Moore.
- Pela simplicidade de implementação e comprovada eficiência, o BM deve ser escolhido em aplicações de uso geral que necessitam realizar casamento exato de cadeias.

Algoritmo BM

- O algoritmo é um dos mais eficientes algoritmos de busca em strings.
- Ele foi desenvolvido em 1977 pelos professores Robert Stephen **Boyer** e J Strother **Moore**.
- Também chamado de Boyer-Moore.
- Pela simplicidade de implementação e comprovada eficiência, o BM deve ser escolhido em aplicações de uso geral que necessitam realizar casamento exato de cadeias.
- A ideia é pesquisar no padrão no sentido da direita para a esquerda, pois isto torna o algoritmo muito rápido.

Algoritmo BM

- O algoritmo é um dos mais eficientes algoritmos de busca em strings.
- Ele foi desenvolvido em 1977 pelos professores Robert Stephen **Boyer** e J Strother **Moore**.
- Também chamado de Boyer-Moore.
- Pela simplicidade de implementação e comprovada eficiência, o BM deve ser escolhido em aplicações de uso geral que necessitam realizar casamento exato de cadeias.
- A ideia é pesquisar no padrão no sentido da direita para a esquerda, pois isto torna o algoritmo muito rápido.
- O algoritmo é comumente usado por editores de texto para executar os comandos “localizar” e “substituir”.

Algoritmo BM

Algoritmo BM

- Busca por Força Bruta:

Algoritmo BM

- Busca por Força Bruta:
  - Considere a busca de LO em HELLO WORLD.

Algoritmo BM

- Busca por Força Bruta:
  - Considere a busca de LO em HELLO WORLD.

Algoritmo BM

- Busca por Força Bruta:
  - Considere a busca de LO em HELLO WORLD.

  - Qual seria é o melhor caso por força bruta?

Algoritmo BM

- Busca por Força Bruta:
  - Considere a busca de LO em HELLO WORLD.

  - Qual seria é o melhor caso por força bruta?
    - O primeiro caractere do padrão não está presente no texto.

Algoritmo BM

- Busca por Força Bruta:
  - Considere a busca de LO em HELLO WORLD.

  - Qual seria é o melhor caso por força bruta?
    - O primeiro caractere do padrão não está presente no texto.
    - Ex: P = “BOLA” e T = “PISTA DE CORRIDA”

Algoritmo BM

- Busca por Força Bruta:
  - Considere a busca de LO em HELLO WORLD.

  - Qual seria é o melhor caso por força bruta?
    - O primeiro caractere do padrão não está presente no texto.
    - Ex: P = “BOLA” e T = “PISTA DE CORRIDA”
    - Neste caso, a complexidade seria O(n).

Algoritmo BM

Algoritmo BM

- Busca por Força Bruta:

Algoritmo BM

- Busca por Força Bruta:
  - Qual seria é o pior caso por força bruta?

Algoritmo BM

- Busca por Força Bruta:
  - Qual seria é o pior caso por força bruta?
    - Quando o padrão e texto são os mesmos.

Algoritmo BM

- Busca por Força Bruta:
  - Qual seria é o pior caso por força bruta?
    - Quando o padrão e texto são os mesmos.
    - Ex: P = “SSS” e T = “SSSSSSS”.

Algoritmo BM

- Busca por Força Bruta:
  - Qual seria é o pior caso por força bruta?
    - Quando o padrão e texto são os mesmos.
    - Ex: P = “SSS” e T = “SSSSSSS”.
    - Neste caso, a complexidade seria O(m*(n-m+1)).

Algoritmo BM

Algoritmo BM

- O enfoque do algoritmo BM consiste em pesquisar o padrão P em uma janela que desliza ao longo do texto T.

Algoritmo BM

- O enfoque do algoritmo BM consiste em pesquisar o padrão P em uma janela que desliza ao longo do texto T.
- Para cada posição desta janela, o algoritmo faz uma pesquisa por um sufixo da janela que casa com um sufixo de P por meio de comparações realizadas no sentido da direita para a esquerda.

Algoritmo BM

- O enfoque do algoritmo BM consiste em pesquisar o padrão P em uma janela que desliza ao longo do texto T.
- Para cada posição desta janela, o algoritmo faz uma pesquisa por um sufixo da janela que casa com um sufixo de P por meio de comparações realizadas no sentido da direita para a esquerda.
- Se não ocorreu uma desigualdade, então uma ocorrência de P em T foi encontrada.

Algoritmo BM

- O enfoque do algoritmo BM consiste em pesquisar o padrão P em uma janela que desliza ao longo do texto T.
- Para cada posição desta janela, o algoritmo faz uma pesquisa por um sufixo da janela que casa com um sufixo de P por meio de comparações realizadas no sentido da direita para a esquerda.
- Se não ocorreu uma desigualdade, então uma ocorrência de P em T foi encontrada.
- Senão, o algoritmo calcula um deslocamento em que o padrão deve ser deslizado para a direita antes que uma nova tentativa de casamento se inicie.

Algoritmo BM

Algoritmo BM

- O algoritmo usa uma tabela de deslocamento.

Algoritmo BM

- O algoritmo usa uma tabela de deslocamento.
- A tabela segue a fórmula abaixo:

Algoritmo BM

- O algoritmo usa uma tabela de deslocamento.
- A tabela segue a fórmula abaixo:
  - Valor = Max(1,Tamanho do Padrão - Index -1)

Algoritmo BM

- O algoritmo usa uma tabela de deslocamento.
- A tabela segue a fórmula abaixo:
  - Valor = Max(1,Tamanho do Padrão - Index -1)
  - Exemplo:

Algoritmo BM

- O algoritmo usa uma tabela de deslocamento.
- A tabela segue a fórmula abaixo:
  - Valor = Max(1,Tamanho do Padrão - Index -1)
  - Exemplo:

PATTERN	H	E	L	L	O
INDEX	0	1	2	3	4

KEY	H	E	L	O
VALUE	Max(1,5-0-1=4) = 4	Max(1,5-1-1=3) = 3	Max(1,5-3-1=1) = 1	Max(1,5-4-1=0) = 1

Algoritmo BM

Algoritmo BM

- Exemplo:

Algoritmo BM

- Exemplo:
  - Considere P = ‘HELLO’ e T = ‘LO LOELLO O HELLO’.

Algoritmo BM

- Exemplo:
  - Considere P = ‘HELLO’ e T = ‘LO LOELLO O HELLO’.

Algoritmo BM

Algoritmo BM

- Neste método, nós pré-processamos o padrão e armazenamos a última ocorrência de todo possível caractere em um vetor de tamanho igual ao tamanho do alfabeto.

Algoritmo BM

- Neste método, nós pré-processamos o padrão e armazenamos a última ocorrência de todo possível caractere em um vetor de tamanho igual ao tamanho do alfabeto.
- Se o caractere não estiver presente, então isso pode resultar em um deslocamento de tamanho igual ao padrão.

Algoritmo BM

- Neste método, nós pré-processamos o padrão e armazenamos a última ocorrência de todo possível caractere em um vetor de tamanho igual ao tamanho do alfabeto.
- Se o caractere não estiver presente, então isso pode resultar em um deslocamento de tamanho igual ao padrão.
- Portanto, no melhor caso, a complexidade seria O(n/m), onde m é o tamanho do padrão.

Algoritmo BM

- Neste método, nós pré-processamos o padrão e armazenamos a última ocorrência de todo possível caractere em um vetor de tamanho igual ao tamanho do alfabeto.
- Se o caractere não estiver presente, então isso pode resultar em um deslocamento de tamanho igual ao padrão.
- Portanto, no melhor caso, a complexidade seria O(n/m), onde m é o tamanho do padrão.
- O pior caso ocorre quando todos os caracteres do padrão e do texto são os mesmos. Ex: “SSS” em “SSSSSSSSSS”. Neste caso, a complexidade seria O(n.m).

Algoritmo BM

Algoritmo BM

- A maioria dos algoritmos de busca em strings requer que o texto seja pré-processado.

Algoritmo BM

- A maioria dos algoritmos de busca em strings requer que o texto seja pré-processado.
- Isto será “caro”, se você tiver que refazer isso frequentemente.

Algoritmo BM

- A maioria dos algoritmos de busca em strings requer que o texto seja pré-processado.
- Isto será “caro”, se você tiver que refazer isso frequentemente.
- BM exige apenas que o padrão seja pré-processado.

Algoritmo BM

- A maioria dos algoritmos de busca em strings requer que o texto seja pré-processado.
- Isto será “caro”, se você tiver que refazer isso frequentemente.
- BM exige apenas que o padrão seja pré-processado.
- Isto torna o algoritmo bastante eficiente.

Casamento Aproximado

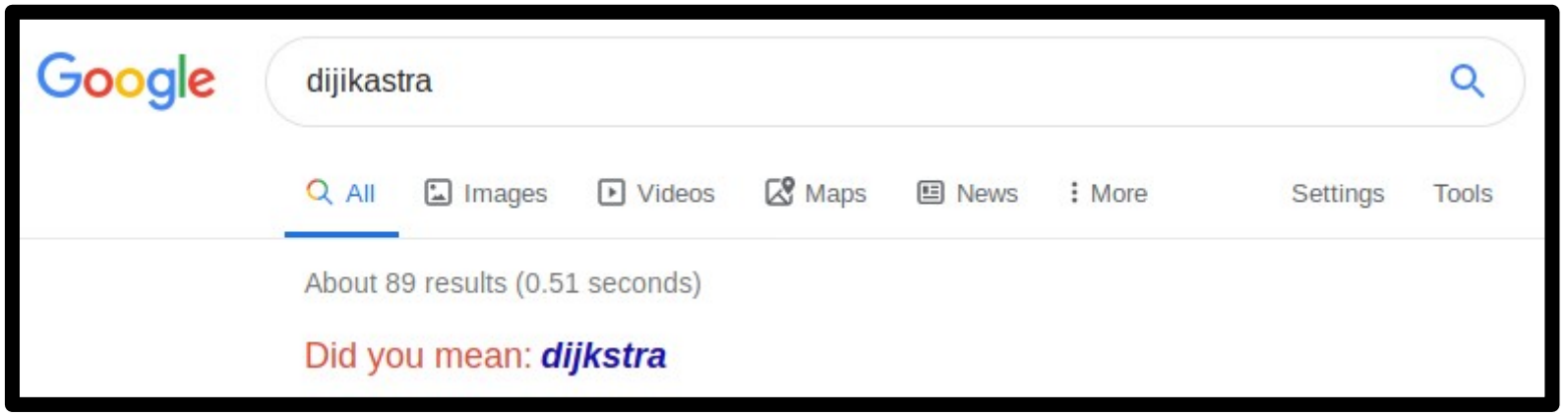
Casamento Aproximado

Casamento Aproximado

- Nem sempre procuramos de forma exata...

Casamento Aproximado

- Nem sempre procuramos de forma exata...



Casamento Aproximado

- Nem sempre procuramos de forma exata...

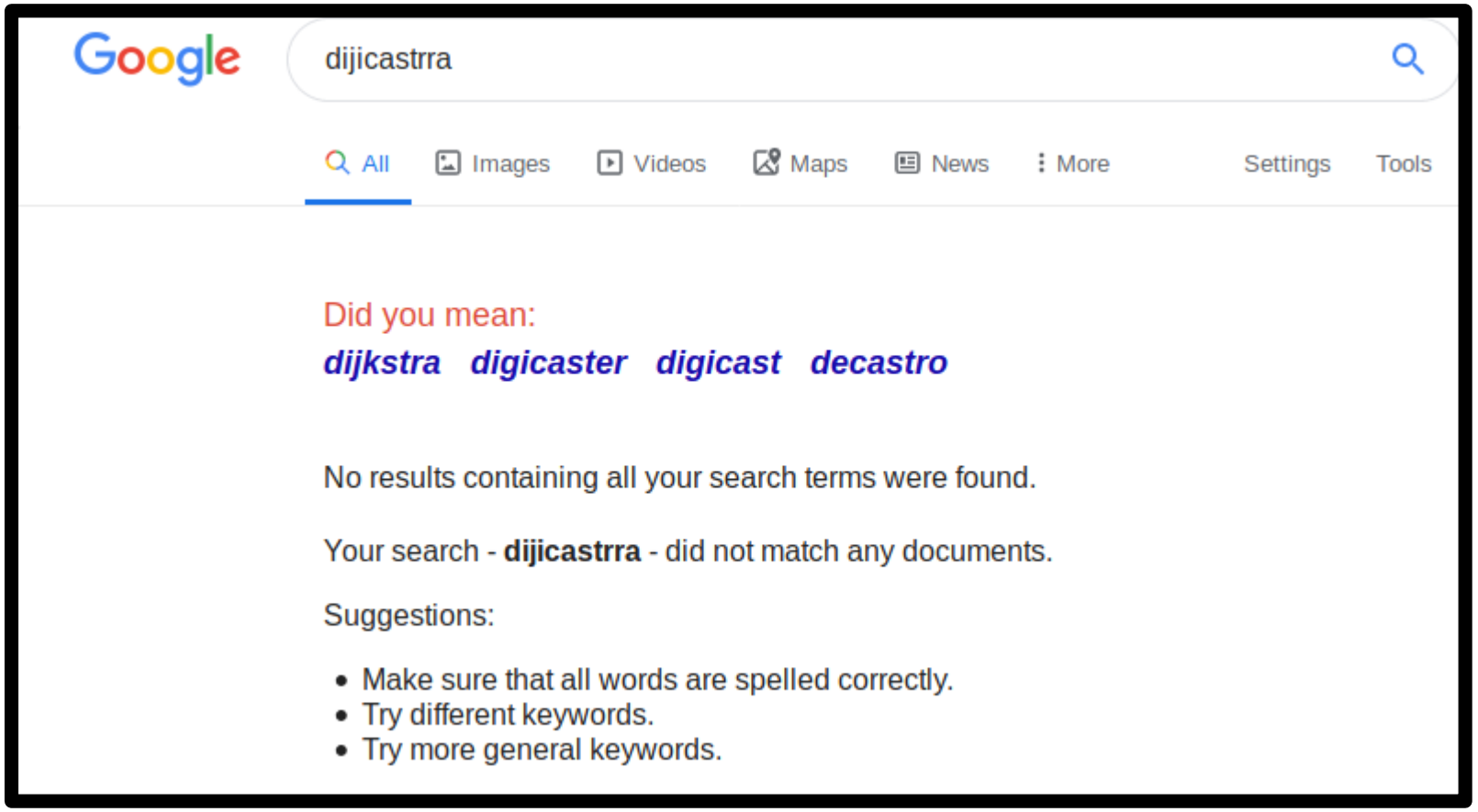
Casamento Aproximado

Casamento Aproximado

- E, às vezes, o Google apenas recomenda...

Casamento Aproximado

- E, às vezes, o Google apenas recomenda...



Casamento Aproximado

Casamento Aproximado

- Existem variações com relação ao casamento aproximado.

Casamento Aproximado

- Existem variações com relação ao casamento aproximado.
- A mais importante é aquela que permite “alterações” no que você procura.

Casamento Aproximado

- Existem variações com relação ao casamento aproximado.
- A mais importante é aquela que permite “alterações” no que você procura.
- Por exemplo, ao procurar a palavra *este*, talvez pesquisar também por *esse*, *essa*, *esta*, *isso*, *nessa*…

Casamento Aproximado

Casamento Aproximado

- Distância de edição

Casamento Aproximado

- Distância de edição
  - É o número de operações (inserção, substituição ou retirada de caracteres) para transformar uma string X em uma string Y.

Casamento Aproximado

- Distância de edição
  - É o número de operações (inserção, substituição ou retirada de caracteres) para transformar uma string X em uma string Y.
  - Exemplos:

Casamento Aproximado

- Distância de edição
  - É o número de operações (inserção, substituição ou retirada de caracteres) para transformar uma string X em uma string Y.
  - Exemplos:
    - Distância = 1 para “este” e “esse”

Casamento Aproximado

- Distância de edição
  - É o número de operações (inserção, substituição ou retirada de caracteres) para transformar uma string X em uma string Y.
  - Exemplos:
    - Distância = 1 para “este” e “esse”
    - Distância = 3 para “este” e “isso”

Casamento Aproximado

Casamento Aproximado

- Algoritmos

Casamento Aproximado

- Algoritmos
  - Não são triviais.

Casamento Aproximado

- Algoritmos
  - Não são triviais.
  - Basicamente são modificações de algoritmos conhecidos:

Casamento Aproximado

- Algoritmos
  - Não são triviais.
  - Basicamente são modificações de algoritmos conhecidos:
    - Shift-And.

Casamento Aproximado

- Algoritmos
  - Não são triviais.
  - Basicamente são modificações de algoritmos conhecidos:
    - Shift-And.
    - Algoritmos baseados em AF.

Casamento Aproximado

- Algoritmos
  - Não são triviais.
  - Basicamente são modificações de algoritmos conhecidos:
    - Shift-And.
    - Algoritmos baseados em AF.
  - Não serão abordados nessa disciplina.

Exercícios

- Considere a implementação abaixo do algoritmo KMP e comente cada linha.

```
1 def KMPSearch(pat, txt):
2         M = len(pat)
3         N = len(txt)
4         lps = [0]*M
5         j = 0
6         computeLPSArray(pat, M, lps)
7         i = 0
8         while i < N:
9                 if pat[j] == txt[i]:
10                        i += 1
11                        j += 1
12
13                if j == M:
14                        print "Padrao encontrado em " + str(i-j)
15                        j = lps[j-1]
16
17                elif i < N and pat[j] != txt[i]:
18                        if j != 0:
19                                j = lps[j-1]
20                        else:
21                                i += 1
22
23 def computeLPSArray(pat, M, lps):
24        len = 0
25        lps[0]
26        i = 1
27        while i < M:
28                if pat[i]== pat[len]:
29                        len += 1
30                        lps[i] = len
31                        i += 1
32                else:
33                        if len != 0:
34                                len = lps[len-1]
35                        else:
36                                lps[i] = 0
37                                i += 1
38
39 txt="ABC ABCDAB ABCDABCDABDE"
40 pat = "ABABACA"
41 KMPSearch(pat, txt)
```

Exercícios

- Por que o algoritmo BM é comumente escolhido por programas de edição de texto no lugar do algoritmo KMP?
- Considerando o texto e o padrão definidos abaixo, utilize os algoritmos estudados para descobrir todas as ocorrências do padrão no texto. Para cada método, indique a quantidade de comparações efetuadas.
  - P = GCAGAGAG
  - T = GCATCGCAGAGAGTATACAGTACG