UNIVERSIDADE FEDERAL DE OURO PRETO – UFOP

Ciência da Computação

ESTRUTURA DE DADOS II

RESUMO CASAMENTO DE CADEIAS

Gabriel Mace do Santos Ferreira

Marcus Vinícius Souza Fernandes

Ouro Preto

2021

Introdução ao casamento de cadeias

O casamento de cadeias consiste em encontrar todas as ocorrências de um determinado padrão em um texto, como em: aplicativos de edição de texto ou durante a recuperação de uma informação. É importante reforçar que a cadeia referenciada no método é uma sequência de elementos denominados caracteres que são escolhidos a partir de um conjunto denominado alfabeto, ou seja, os elementos que iram formar o alfabeto.

Exemplo: Em uma cadeia de bits o alfabeto é composto pelos valores {0,1}.

Elaboração do problema:

- Texto: Cadeia T[0 … n-1] de tamanho **n**.
- Padrão: Cadeia P[0 … m-1] de tamanho **m** maior ou igual a **n**.
- Os caracteres de P e T são escolhidos a partir de um alfabeto finito de tamanho **c** (Exemplo: A = {0,1} ou A = {a,b, … ,z}).
- O casamento de cadeias ocorre quando, dadas as cadeias P e T, deseja-se saber as ocorrências de P em T.

P e T não são pré-processados: Ou seja, tanto o padrão quando o texto não são conhecidos a priori. O algoritmo é sequencial, online, de tempo real e possui uma complexidade de tempo de O(**mn**) e uma complexidade de espaço de O(1).

P é processado: Ou seja, o padrão é conhecido a priori. O algoritmo é sequencial, possuindo uma complexidade de tempo de O(**n**) e complexidade de espaço de O(**m+c**).

Algoritmo em que P e T são pré-processados: Onde P e T são conhecidos a priori e o algoritmo constrói índice para texto. Este índice varia, podendo ser O(**n**) ou O(**n log n**), no entanto ocorre uma compensação desse tempo por meio de muitas operações de pesquisa no texto, tornando a complexidade de tempo do algoritmo O(**log n**) e a complexidade de espaço O(**n**).

Tipos de índices:

- Arquivos invertidos.
- Árvores TRIE e árvores PATRICIA.
- Arranjos de sufixos.

Arquivo invertido

Podemos ressaltar que um arquivo invertido é composto por duas partes, vocabulário e ocorrências, onde o vocabulário se define como o conjunto de palavras variadas contidas no texto em questão e para cada palavra diferente, uma lista de posições onde a mesma é localizada é armazenada em uma lista, este conjunto de listas se caracteriza como as ocorrências.

O vocabulário ocupa menos espaço em relação às ocorrências, sua previsão de crescimento é definida pela lei de Heaps onde:

- Um determinado texto em linguagem natural contendo **n** palavras de tamanho **v = Kn^beta =** O(**n^beta**), sendo **K** e **beta** dependentes das características do texto em questão.
- **K** geralmente possui valores no intervalo de 10 à 100 e **beta** é uma constante que pode variar de 0 à 1 (na prática 0,4 e 0,6).
- Podemos destacar que o vocabulário cresce com o tamanho do texto, em uma proporção próxima de sua raiz quadrada.

As ocorrências ocupam mais espaço, ressaltando:

- O espaço necessário é O(**n**) já que cada palavra é referenciada apenas uma vez na lista de ocorrências.
- Na prática, o espaço fica entre 30% a 50% do tamanho do texto original.

Etapas da pesquisa, os três passos:

1. Pesquisa no vocabulário: As palavras da consulta são isoladas e pesquisadas no vocabulário. Dado que a pesquisa se inicia pelo vocabulário é indicado mantê-lo em um arquivo separado que usualmente cabe na memória principal.
2. Recuperação de ocorrências: As listas de ocorrências das palavras encontradas no vocabulário são recuperadas.
3. Manipulação das ocorrências: As listas de ocorrências são processadas para tratar frases, proximidade e/ou operações lógicas.

O comportamento do método de casamento de cadeias do arquivo invertido pode diferir dependendo da situação, visto que frente a pesquisa de palavras simples, qualquer estrutura de dados que torne a pesquisa eficiente pode ser utilizada (Exemplo: Hashing, árvore TRIE ou árvore B), no entanto a pesquisa de frases utilizando índices é mais complexa sendo que cada palavra deve ser pesquisada de forma individual.

A fim de recuperar as listas de recorrência, posteriormente as listas devem ser percorridas sincronizadamente no intuito de encontrar uma ocorrência em que as palavras apareçam em sequência, formando a frase desejada.

Inverted Index Example

ID	Text
1	Baseball is played during summer months.
2	Summer is the time for picnics here.
3	Months later we found out why.
4	Why is summer so hot here

↑ Sample document data

Dictionary and posting lists →

Term	Freq	Document ids
baseball	1	[1]
during	1	[1]
found	1	[3]
here	2	[2], [4]
hot	1	[4]
is	3	[1], [2], [4]
months	2	[1], [3]
summer	3	[1], [2], [4]
the	1	[2]
why	2	[3], [4]

Casamento Exato

A dificuldade de realizar um casamento exato de cadeias ocorre, devido a necessidade de encontrar as ocorrências exatas de um padrão em um determinado texto.

Categorização de algoritmos: Evidenciando que a identificação de um algoritmo é baseada na forma como o padrão é pesquisado no texto.

1. Leitura dos caracteres do texto T um à um, no intuito de identificar uma ocorrência possível do padrão P (Exemplo: Algoritmos força bruta e Shift-And).
2. Pesquisa do padrão P em uma janela que desliza ao longo do texto (T), procurando por um sufixo de T que após comparações da direita para esquerda casa com o sufixo(P) (Exemplo: Algoritmos Boyer-Moore, Boyer-Moore-Horspool e o Boyer-Moore-Horspool-Sunday).

Algoritmo Força Bruta

O algoritmo de força bruta se destaca por ser o algoritmo mais simples para o casamento exato de cadeias. A sua função consiste em casar qualquer subcadeia de comprimento **m** no texto com o padrão desejado.

- Pior caso: $\mathbf{C_n = m * n}$
- Caso esperado: $\mathbf{C_n = (c/c-1) * (1-(1/(c^m))) * (n-m+1) + O(1)}$

Algoritmo Boyer-Moore

O algoritmo de Boyer-Moore pesquisa o padrão P em uma janela que desliza ao longo do texto T. Para cada posição da janela é pesquisado um sufixo que casa com o sufixo de P, as comparações são realizadas no sentido da direita para a esquerda.

Uma ocorrência de P em T foi encontrada se não ocorrer uma desigualdade, caso contrário o algoritmo calcula o quanto a janela deve deslizar para a direita antes que uma nova tentativa de casamento se inicie.

O cálculo do deslocamento pode ser feito por meio de duas heurísticas:

1. Heurística de ocorrência: A ocorrência alinha o caractere em T que causou a colisão com o primeiro caractere de P, a esquerda do ponto de colisão, que se casa com ele.

2. Heurística do casamento: Ao mover o padrão para a direita, a janela em questão casa com o trecho do texto anteriormente casado.

A escolha de qual heurística utiliza é feita baseado em qual delas irá provocar o maior deslocamento do padrão. Sendo necessário realizar comparações para cada colisão que ocorrer, penalizando o desempenho do algoritmo com relação ao tempo de processamento. Para alfabetos relativamente grandes, utiliza-se a heurística de ocorrência, já para alfabetos menores a heurística de casamento se torna mais recomendada.

Vale notar que ao longo dos anos diversas propostas para simplificação surgiram e os melhores resultados foram obtidos com a utilização da heurística de ocorrência.

Algoritmo Boyer-Moore-Horspool

O algoritmo de Boyer-Moore-Horspool foi desenvolvido por Horspool em 1980 como uma simplificação do algoritmo de Boyer-Moore, tornando o mais rápido e simples, devido a isso o algoritmo BMH é indicado em aplicações de uso geral que necessitam do casamento exato de cadeias.

A partir de qualquer caractere já lido em T no último deslocamento é possível endereçar uma tabela de deslocamentos. Horspool propôs que o deslocamento da janela deveria ser feito de acordo com o valor da tabela de deslocamento relativo ao caractere no texto correspondente ao último caractere do padrão.

A fim de definir a tabela de deslocamento é necessário que o valor inicial do deslocamento para todos os caracteres do texto seja igual a **m**. Posteriormente para os **m-1** primeiros caracteres do padrão P, os valores do deslocamento são calculados pela regra:

d[x] = min { j tal que (j = m) | (1 <= j < m & P[m-j] = x) }

Algoritmo Boyer-Moore-Horspool-Sunday

O Boyer-Moore-Horspool-Sunday foi apresentado em 1990 como uma simplificação importante para o algoritmo BMH.

A sua simplificação consiste em:

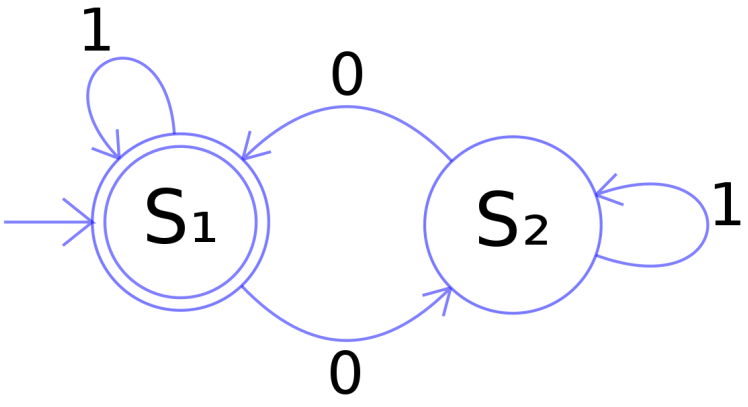
- Ser uma variante do algoritmo BMH.
- O mesmo desloca a janela de acordo com o valor contido na tabela de deslocamento relativo ao caractere do texto correspondente ao caractere que se encontra após o último caractere do padrão proposto.

Para se definir a tabela de deslocamento, faz-se:
O valor inicial para o deslocamento de todos os caracteres é **m+1**.
O passo seguinte, para os m primeiros caracteres do padrão dado (P), os valores do deslocamento são calculados pela seguinte regra:

d[x] = min{ j tal que (j = m+1) | (1 <= j <= m & P[m+1-j] = x) }

Autômatos

Um autômato é um modelo de computação simples, que pode ser dividido em diversas subcategorias.



Autômatos Finitos:

Um autômato finito é definido pela tupla (Q, I, F, ∑, T), sendo que:

- **Q** é um conjunto de estados

- **I** é o estado inicial **(I ∈ Q)**
- **F** é o conjunto de estados finais **(F ⊆ Q)**
- ∑ é o alfabeto finito de entrada
- **T** é a função que define as transições entre os estados, além **T** associa a cada estado **q ∈ Q** um conjunto de estados **{q_1, q_2, …., q_k}** **⊆ Q**, para cada **α ∈ (∑ ∪ {ε})**, onde **ε** é a transição vazia

Uma cadeia é reconhecida pelo autômato finito se esse rotula um caminho, que vai do estado inicial até um estado final, compreendendo a cadeia em questão. Vale ressaltar que a linguagem reconhecida por um autômato é o conjunto de cadeias que o autômato é capaz de reconhecer.

A transição vazia representada pelo símbolo **ε,** conhecida como transição-**ε** em autômatos não-deterministas. Esse tipo de transição simplifica a construção do autômato dado que não é necessário ler um caractere do alfabeto para se caminhar por meio de uma transição vazia. É importante destacar que sempre existe um autônomo equivalente que reconhece a mesma linguagem sem transições vazias.

Um estado **q** é considerado ativo quando uma cadeia x rotula um caminho de **I** até esse estado.

Autômato Finito Não-Determinista:

Quando a função **T** possibilita a associação de um estado **q** e um caractere α para mais de um estado do autômato (T(q,α) = {q_1, q_2, …., q_k} para **k** > 1), ou quando existe uma transição rotulado por ε.

É possível inferir que um autômato não-determinista pode ter vários estados ativos em um determinado instante,permitindo a resolução do casamento aproximado de cadeias por meio desse autômato.

Autômato Finito Determinista:

Inversamente ao autômato finito não-determinista, esta ocorre quando a função T permite a associação de um estado **q** e um caractere α para apenas um estado do autômato (T(q,α) = {q_1}).

Devido às propriedades desse autômato ele pode possuir, no máximo, apenas um estado ativo em um determinado instante.

Autômato Acíclico:

Um autômato acíclico ocorre quando suas transições não formam ciclos.

Autômato Cíclico:

Um autômato cíclico ocorre quando suas transições formam ciclos. Nos autômatos finitos a formação desses ciclos é útil para o casamento de expressões regulares.

Em um autômato cíclico a linguagem reconhecida por esse pode ser infinita.

Exemplo: Dado um padrão P = {aabc}. A pesquisa desse padrão em um texto com alfabeto Σ ={a, b, c}, pode ser compreendida como a simulação do autômato na pesquisa de P sobre T. Essa pesquisa possui uma complexidade de O(n), dado que cada caractere do texto é lido pelo menos uma vez, ademais a complexidade de espaço é dada por (m+1) para os vértices que compõem as arestas.

Algoritmo Shift-And Exato

O algoritmo Shift-And Exato utiliza o conceito de paralelismo de bit, onde:

- Consiste em uma técnica que tira proveito do paralelismo das operações sobre bits dentro de uma palavra do computador, possibilitando "empacotar" muitos valores de uma só vez e atualizá-los com uma única operação realizada posteriormente.
- Uma sequência de bits que se enquadra no intervalo de **b1** à **bc** é denominada de máscara de bits de comprimento **c,** é armazenada em alguma posição de palavra **w** da máquina.

Operações sobre bits de uma palavra:

- Repetição: Exponenciação (Exemplo: 01^3 = 0111).
- Operador lógico OU: "**|**".
- Operador lógico E: "**&**".
- Operador que move à direita e entra com zeros a esquerda: "**>>**".

Uma das características do algoritmo é que o mesmo mantém o conjunto de todos os prefixos do padrão P que casam com o texto já lido. O Shift-And também corresponde à uma simulação de um Autômato (contextualizado anteriormente) Não-Determinista que pesquisa pelo padrão P no texto em questão T.

Nos passos do algoritmo, o valor 1 é colocado na j-ésima posição de R = (**b1** … **bm**) se somente se (**p1** … **pj**) for um sufixo de (**t1** … **tj**), onde **i** corresponde à posição corrente no texto em questão. O **bm** ativo significa um casamento exato do padrão desejado.

R' corresponde ao novo valor da máscara **R**, ele é calculado na leitura do próximo caractere **t(i+1)** do texto.

- A posição **j+1** em **R'** ficará ativa se somente a posição **j** estava ativa em **R**, ou seja, (**p1** … **pj**) é um sufixo de (**t1** … **tj**) e **t(i+1)** se casa com **p(j+1)**.

O pré processamento do algoritmo consiste na construção de uma tabela M para armazenar uma máscara de bits para cada um dos caracteres do padrão. Onde os caracteres são marcados pelo valor **1** caso se apresentem na posição correta ou pelo valor **0** em caso de não haver esta compatibilidade.

Exemplo:

	1	2	3	4	5
M[‘A’]	1	0	0	0	1
M[‘J’]	0	1	0	0	0
M[‘U’]	0	0	1	0	0
M[‘D’]	0	0	0	1	0

A máscara de bits **R** é sempre inicializada como **R = 0^m**. E para cada novo caractere **t(i+1)** lido do texto, o valor de **R’** é atualizado pela seguinte expressão:

R’ = ((R >> 1) | 10^(m-1)) & M[T[i]]

Obs: Todas as operações realizadas na função acima foram descritas anteriormente.

Em suma podemos analisar o custo do algoritmo como O(**n**), desde que as operações possam ser realizadas em O(1) e o padrão possa ser alocado em poucas palavras na máquina em questão.

Casamento aproximado

A complexidade do casamento aproximado de cadeias ,se dá na identificação das ocorrências aproximadas de um padrão (P’) no texto (T), por isso devem ser tratadas operações de inserção, substituição e retirada dos caracteres do padrão.

O tratamento dessas operações é efetuado ao comparar a distância de edição entre P e P’ com o limite de operações de inserção, retirada e substituição necessárias para transformar P em P’.

É importante destacar que a distância de edição entre duas cadeias (**ed**(P, P’)) é o menor número de operações necessárias para converter P em P’ ou vice-versa.

O uso do casamento aproximado é recomendado quando 0 < **k** < **m** , visto que para **k** = **m**, toda subcadeia de comprimento m pode ser convertida em P por meio da substituição de **m** caracteres.

Obs: k = 0 corresponde ao casamento exato de cadeias.

Os autômatos não-deterministas são utilizados na pesquisa com casamento aproximado.

Nível de erro:

- O nível de erro é dado por **α = k/m**
- Usualmente **α** < 1/2

Algoritmo Shift-And Aproximado

O algoritmo Shift-And Aproximado simula um autômato não-determinista e utiliza paralelismo de bit. Ele"empacota" cada linha **j** (0 < **j** <= **k**) em uma palavra **Rj** diferente do computador.

Características:

- A máscara **R0** (casamento exato) é sempre inicializada como **R0 = 0^m**.
- Para 0 < **j** <= **k**, **Rj** é inicializado como **Rj = 1^(j)0^(m-j)**.
- Considerando M a tabela do algoritmo Shift-And para casamento exato (evidenciada anteriormente), para cada novo caractere **t(i+1)** lido do texto, as máscaras são atualizadas por:

R'0 = ((R0 >> 1) | 10^(m-1)) & M[T[i]] num intervalo de 0 < **j** <= **k**

R'j = ((Rj >> 1) & M[T[i]]) | R(j-1) | (R(j-1) >> 1) | (R'(j-1) >> 1) | 10^(m-1)

- Considerando o autômato, a fórmula **R'** expressa as arestas:

  - Horizontais: Indicando um casamento.
  - Verticais: Indicando inserção **(R(j-1))**.
  - Diagonais cheias: Indicando substituição **(R(j-1) >> 1)**.
  - Diagonais tracejadas: Indicando substituição **(R'(j-1) >> 1)**.

A máscara de bits é atualizada pelas seguintes expressões:

R'0 = ((R0 >> 1) | 10^(m-1)) & M[T[i]]
R'1 = ((R1 >> 1) & M[T[i]]) | R0 | 10^(m-1)
R'1 = ((R1 >> 1) & M[T[i]]) | R0 | (R0 >> 1) | (R'0 >> 1) | 10^(m-1)