Estrutura de dados 1

Processamento de Cadeias de Caracteres

Casamento de Cadeias

- Casamento de Cadeias
  - Casamento Exato
  - Casamento Aproximado
- Compressão
  - Por Que Usar Compressão
  - Compressão de Textos em Linguagem Natural
  - Codificação de Huffman Usando Palavras

Definição e Motivação

- Cadeia de caracteres: sequência de elementos denominados caracteres.
- Os caracteres são escolhidos de um conjunto denominado alfabeto.
  - Ex.: em uma cadeia de bits o alfabeto é {0, 1}.
- Casamento de cadeias de caracteres ou casamento de padrão:
  - encontrar todas as ocorrências de um padrão em um texto.
  - Exemplos de aplicação:
    - edição de texto;
    - recuperação de informação;
    - estudo de sequências de DNA em biologia computacional.

Notação

- Texto: arranjo T [1..n] de tamanho n;
- Padrão: arranjo P [1..m] de tamanho $m \leq n$.
- Os elementos de P e T são escolhidos de um alfabeto finito Σ de tamanho c.
  - Ex.: $\Sigma = \{0,1\}$ ou $\Sigma = \{a,b,...,z\}$.
- **Casamento de cadeias** ou **casamento de padrão**: dados duas cadeias P (padrão) de comprimento $|P| = m$ e T (texto) de comprimento $|T| = n$, onde $n \gg m$, deseja-se saber as ocorrências de P em T.

Categorias de Algoritmos

- P e T não são pré-processados:
  - algoritmo sequencial, on-line e de tempo-real;
  - padrão e texto não são conhecidos *a priori*.
  - complexidade de tempo O(mn) e de espaço O(1), para pior caso.
- P pré-processado:
  - algoritmo sequencial;
  - padrão conhecido anteriormente permitindo seu pré-processamento.
  - complexidade de tempo O(n) e de espaço O(m + c), no pior caso.
    - ex.: programas para edição de textos.

Categorias de Algoritmos

- P e T são pré-processados:
  - algoritmo constrói índice.
  - complexidade de tempo O(log n) e de espaço é O(n).
  - tempo para obter o índice é grande, O(n) ou O(n log n).
  - compensado por muitas operações de pesquisa no texto.
  - Tipos de índices mais conhecidos:
  - Arquivos invertidos
  - Árvores *trie* e árvores Patricia
  - Arranjos de sufixos

Exemplos: P e T são pré-processados

- Diversos tipos de índices: arquivos invertidos, árvores *trie* e Patricia, e arranjos de sufixos.
- Um **arquivo invertido** possui duas partes:
  - **vocabulário** e **ocorrências**.
  - O vocabulário é o conjunto de todas as palavras distintas no texto.
    - Para cada palavra distinta, uma lista de posições onde ela ocorre no texto é armazenada.
  - O conjunto das listas é chamado de ocorrências.
  - As posições podem referir-se a palavras ou caracteres.

Exemplo de Arquivo Invertido

1 7 16 22 26 36 45 53
Texto exemplo. Texto tem palavras. Palavras exercem fascínio.

- exemplo 7
- exercem 45
- fascínio 53
- palavras 26 36
- tem 22
- texto 1 16

Arquivo Invertido – Pesquisa

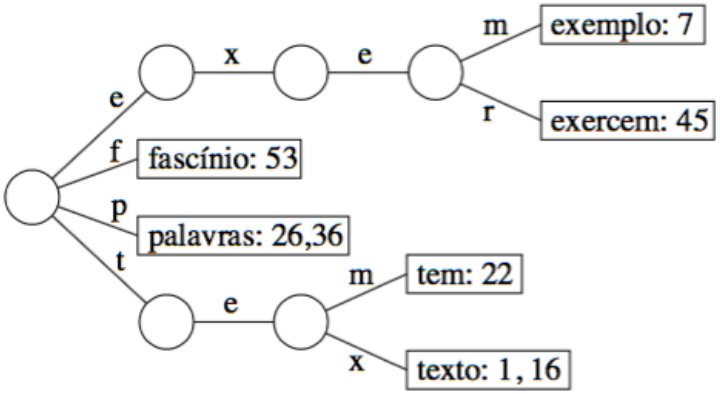
- A pesquisa tem geralmente três passos:
  1. Pesquisa no vocabulário:
     1. palavras e padrões da consulta são isoladas e pesquisadas no vocabulário.
  2. Recuperação das ocorrências:
     1. as listas de ocorrências das palavras encontradas no vocabulário são recuperadas.
  3. Manipulação das ocorrências:
     1. as listas de ocorrências são processadas para tratar frases, proximidade, ou operações booleanas.
- Como a pesquisa em um arquivo invertido sempre começa pelo vocabulário, é interessante mante-lo em um arquivo separado.
- Na maioria das vezes, esse arquivo cabe na memória principal.

Arquivo Invertido – Pesquisa

- A pesquisa de palavras simples pode ser realizada usando qualquer estrutura de dados que torne a busca eficiente, como *hashing*, árvore *trie* ou árvore B.
  - As duas primeiras têm custo O(m), onde m é o tamanho da consulta (independentemente do tamanho do texto).
- Guardar as palavras na ordem lexicográfica é barato em termos de espaço e competitivo em desempenho, já que a pesquisa binária pode ser empregada com custo O(log n).
- A pesquisa por frases usando índices é mais difícil de resolver.
  - Cada elemento da frase tem de ser pesquisado separadamente e suas listas de ocorrências recuperadas.
  - A seguir, as listas têm de ser percorridas de forma sicronizada para encontrar as posições nas quais todas as palavras aparecem em sequência.

Arquivo Invertido Usando Trie

Arquivo invertido usando uma **árvore *trie*** para o texto:
Texto exemplo. Texto tem palavras. Palavras exercem fascínio.



O vocabulário lido até o momento é colocado em uma árvore *trie*, armazenando uma lista de ocorrências para cada palavra.

Arquivo Invertido Usando Trie

- Cada nova palavra lida é pesquisada na *trie*:
  - Se a pesquisa é sem sucesso, então a palavra é inserida na árvore e uma lista de ocorrências é inicializada com a posição da nova palavra no texto.
  - Senão, uma vez que a palavra ja□ se encontra na árvore, a nova posição é inserida ao final da lista de ocorrências.

Casamento Exato

- Consiste em obter todas as ocorrências **exatas** do padrão no texto.
  - Ex.: ocorrência exata do padrão teste.

    teste

    os testes testam estes alunos . . .

- Dois enfoques:
  - leitura dos caracteres do texto um a um:
    - algoritmos força bruta, Knuth-Morris-Pratt e Shift-And.
  - pesquisa de P em uma janela que desliza ao longo de T , pesquisando por um sufixo da janela que casa com um sufixo de P , por comparações da direita para a esquerda:
    - algoritmos Boyer-Moore-Horspool e Boyer-Moore.

Força Bruta - Implementação

- É o algoritmo mais simples para casamento de cadeias.
- A idéia é tentar casar qualquer subcadeia no texto de comprimento m com o padrão.

```
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("Casamentonaposicao%3ld\n", i);
            }
        }
```

Casamento Aproximado

- O casamento aproximado de cadeias permite operações de inserção, substituição e retirada de caracteres do padrão.
  - Ex.: Três ocorrências do padrão teste em que os casos de inserção, substituição, retirada de caracteres no padrão acontecem:
  1. um espaço é inserido entre o terceiro e quarto caracteres do padrão;
  2. o último caractere do padrão é substituído pelo caractere a;
  3. o primeiro caractere do padrão é retirado.

```
tes te
    testa
              este
os testes testam estes alunos . . .
```

Distância de Edição

- Número k de operações de inserção, substituição e retirada de caracteres necessário para transformar uma cadeia x em outra cadeia y.
  - ed(P, P'): distância de edição entre duas cadeias P e P'; é o menor número de operações necessárias para converter P em P', ou vice versa.
  - Ex.: ed(teste,estende) = 4, valor obtido por meio de uma retirada do primeiro t de P e a inserção dos 3 caracteres nde ao final de P .
- O problema do casamento aproximado de cadeias é o de encontrar todas as ocorrências em T de cada P' que satisfaz ed(P, P') $\leq$ k.

Casamento Aproximado

- Abusca aproximada 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.
- O caso em que k = 0 corresponde ao casamento exato de cadeias.
- O nível de erro $\alpha = k/m$, fornece uma medida da fração do padrão que pode ser alterado.
  - Em geral $\alpha < 1/2$ para a maioria dos casos de interesse.
- **Casamento aproximado de cadeias**, ou **casamento de cadeias permitindo erros**:
  - um número limitado k de operações (erros) de inserção, de substituição e de retirada é permitido entre P e suas ocorrências em T .
- A pesquisa com casamento aproximado é modelado por autômato não-determinista.

Compressão - Motivação

- Explosão de informação textual disponível *on-line*:
  - Bibliotecas digitais.
  - Sistemas de automação de escritórios.
  - Bancos de dados de documentos.
  - World-Wide Web.
- Somente a Web tem hoje bilhões de páginas estáticas disponíveis.
  - Cada bilhão de páginas ocupando aproximadamente 10 *terabytes* de texto corrido.
  - Em setembro de 2003, a máquina de busca Google (www.google.com.br) dizia ter mais de 3,5 bilhões de páginas estáticas em seu banco de dados.

Características necessárias para sistemas de recuperação de informação

- Métodos recentes de compressão têm permitido:
  - Pesquisar diretamente o texto comprimido mais rapidamente do que o texto original.
  - Obter maior compressão em relação a métodos tradicionais, gerando maior economia de espaço.
  - Acessar diretamente qualquer parte do texto comprimido sem necessidade de descomprimir todo o texto desde o início (Moura, Navarro, Ziviani e Baeza-Yates, 2000; Ziviani, Moura, Navarro e Baeza-Yates, 2000).
- Compromisso espaço X tempo
  - Melhora nos dois.

Porque Usar Compressão

- **Compressão de texto -** maneiras de representar o texto original em menos espaço
  - Substituir os símbolos do texto por outros que possam ser representados usando um número menor de *bits* ou *bytes*.
- **Ganho obtido:** o texto comprimido ocupa menos espaço de armazenamento ⇒ menos tempo para ser lido do disco ou ser transmitido por um canal de comunicação e para ser pesquisado.
- **Preço a pagar:** custo computacional para codificar e decodificar o texto.
- **Avanço da tecnologia:** De acordo com Patterson e Hennessy (1995), em 20 anos, o tempo de acesso a discos magnéticos tem se mantido praticamente constante, enquanto a velocidade de processamento aumentou aproximadamente 2 mil vezes
  - melhor investir mais poder de computação em compressão em troca de menos espaço em disco ou menor tempo de transmissão.

Razão de Compressão

- Definida pela porcentagem que o arquivo comprimido representa em relação ao tamanho do arquivo não comprimido.
  - **Exemplo:** se o arquivo não comprimido possui 100 *bytes* e o arquivo comprimido resultante possui 30 *bytes*, então a razão de compressão é de 30%.
- Utilizada para medir **O ganho em espaço** obtido por um método de compressão.

Outros Importantes Aspectos a Considerar

- Além da economia de espaço, deve-se considerar:
  - Velocidade de compressão e de descompressão.
  - Possibilidade de realizar **casamento de cadeias** diretamente no texto comprimido.
  - Permitir acesso direto a qualquer parte do texto comprimido e iniciar a descompressão a partir da parte acessada:

- **Um sistema de recuperação de informação para grandes coleções de documentos que estejam comprimidos necessitam acesso direto a qualquer ponto do texto comprimido.**

Compressão de Textos em Linguagem Natural

- Um dos métodos de codificação mais conhecidos é o de **Huffman** (1952):
  - Atribui códigos mais curtos a símbolos com frequências altas.
  - Um código único, de tamanho variável, é atribuído a cada símbolo diferente do texto.
  - As implementações tradicionais do método de Huffman consideram caracteres como símbolos.
  - Para aliar as necessidades dos algoritmos de compressão às necessidades dos sistemas de recuperação de informação apontadas acima, deve-se considerar palavras como símbolos a serem codificados.
  - Métodos de Huffman baseados em caracteres comprimem o texto para aproximadamente 60%.
  - Métodos de Huffman baseados em palavras comprimem o texto para valores pouco acima de 25%.

Métodos de Huffman Baseados em Palavras: Vantagens

- Permitem acesso randômico a palavras dentro do texto comprimido.
- Considerar palavras como símbolos significa que a tabela de símbolos do codificador é exatamente o **vocabulário** do texto.
  - Isso permite uma integração natural entre o método de compressão e o arquivo invertido.
- Permitem acessar diretamente qualquer parte do texto comprimido sem necessidade de descomprimir todo o texto desde o início.

Compressão de Huffman Usando Palavras

- Técnica de compressão mais eficaz para textos em linguagem natural.
- O método considera cada palavra diferente do texto como um símbolo.
  - Conta suas frequências e gera um código de Huffman para as palavras.
  - A seguir, comprime o texto substituindo cada palavra pelo seu código.
- Assim, a compressão é realizada em duas passadas sobre o texto:
  - Obtenção da frequência de cada palavra diferente.
  - Realização da compressão.

Forma Eficiente de Lidar com Palavras e Separadores

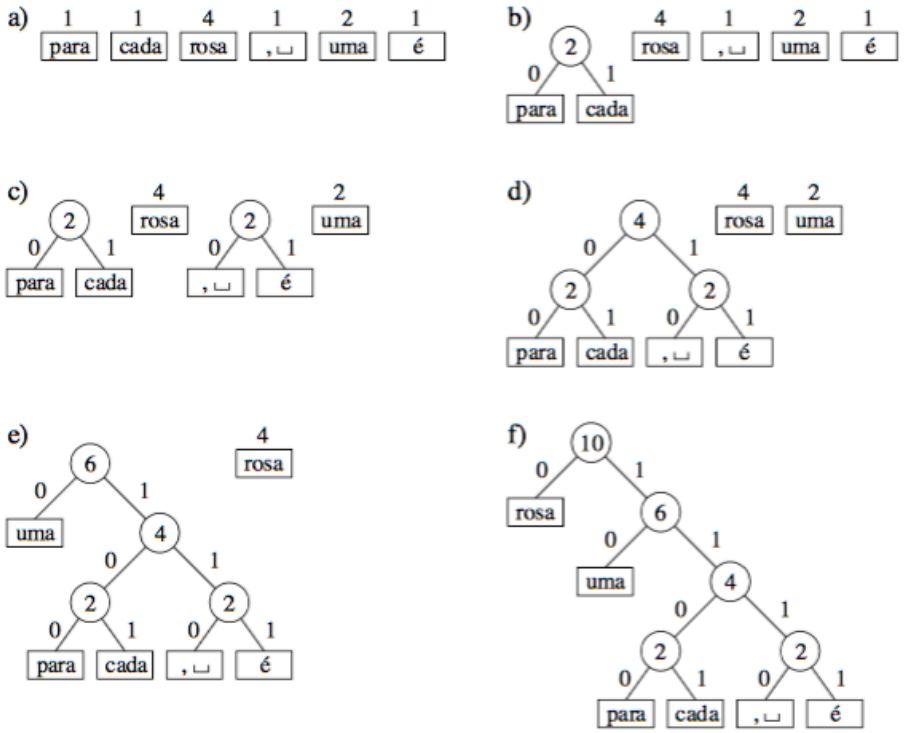
- Um texto em linguagem natural é constituído de palavras e de separadores.
- Separadores são caracteres que aparecem entre palavras:
  - espaço, vírgula, ponto, ponto e vírgula, interrogação, e assim por diante.
- Uma forma eficiente de lidar com palavras e separadores é representar o espaço simples de forma implícita no texto comprimido.
  - Nesse modelo, se uma palavra é seguida de um espaço, então, somente a palavra é codificada.
- Senão, a palavra e o separador são codificados separadamente.
- No momento da decodificação, supõe-se que um espaço simples segue cada palavra, a não ser que o próximo símbolo corresponda a um separador.

Compressão usando codificação de Huffman

Exemplo:

"para cada rosa rosa, uma rosa é uma rosa"

OBS: O algoritmo de Huffman é uma abordagem **gulosa**.



Árvore de Huffman

- O método de Huffman produz a árvore de codificação que minimiza o comprimento do arquivo comprimido.
- Existem diversas árvores que produzem a mesma compressão.
  - Por exemplo, trocar o filho à esquerda de um nó por um filho à direita leva a uma árvore de codificação alternativa com a mesma razão de compressão.
- Entretanto, a escolha preferencial para a maioria das aplicações é a **árvore canônica**.
  - Uma árvore de Huffman é canônica quando a altura da subárvore à direita de qualquer nó nunca é menor que a altura da subárvore à esquerda.

Árvore de Huffman

- A representação do código na forma de árvore facilita a visualização.
- Sugere métodos de codificação e decodificação triviais:
  - **Codificação:** a árvore é percorrida emitindo *bits* ao longo de suas arestas.
  - **Decodificação:** os *bits* de entrada são usados para selecionar as arestas.
  - Essa abordagem é ineficiente tanto em termos de espaço quanto em termos de tempo.

- **Algoritmo Baseado na Codificação Canônica com Comportamento Linear em Tempo e Espaço**
  - O algoritmo é atribuído a Moffat e Katajainen (1995).
  - Calcula os comprimentos dos códigos em lugar dos códigos propriamente ditos.
  - A compressão atingida é a mesma, independentemente dos códigos utilizados.
  - Após o cálculo dos comprimentos, há uma forma elegante e eficiente para a codificação e a decodificação.

Código Canônico

- Os comprimentos dos códigos obedecem ao algoritmo de Huffman.
- Códigos de mesmo comprimento **são** inteiros consecutivos.
- A partir dos comprimentos obtidos, o cálculo dos códigos é trivial:
  - o primeiro código é composto apenas por zeros e,
  - para os demais, adiciona-se 1 ao código anterior e faz-se um deslocamento à esquerda para obter-se o comprimento adequado quando necessário.

i	Símbolo	Código Canônico
1	rosa	0
2	uma	10
3	para	1100
4	cada	1101
5	, _	1110
6	é	1111