Universidade Federal de Goiás
Mestrado em Ciência da Computação
Projeto e Análise de Algoritmos

Profa. Diane

Aluno: Edison Andrade Martins Morais

Tema: **War Story: String'em Up**

Introdução
Caracterização do Problema

- Seqüência do **GENOMA HUMANO**;

- É um padrão (**seqüência DNA**) que codifica toda a informação necessária para construir as proteínas que formam os seres humanos;

Representação de uma **Seqüência DNA:**

- É representada por **Seqüências de Strings** formadas pelos caracteres **A, C, T, G**.

- Estas seqüências são **MUITO LONGAS** (aproximadamente 3 bilhões de caracteres).

Mas,
Qual é o **PROBLEMA?**

- Questão chave:
  - **1. Como encontrar substrings de caracteres?**

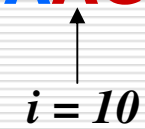


  - **Algoritmo: String Matching**
    - **Entrada:** Texto ***T*** de tamanho **N**, String ***P*** de tamanho **M**.
    - **Problema:** Encontrar a primeira (ou todas) as instâncias de ***P*** em ***T***.
    - **Solução:**
      1. Para textos pequenos
      2. Para textos grandes

Considera-se a **solução sob 2 aspectos**

Qual é a **SOLUÇÃO?**

- **1. Para textos pequenos**
  - Algoritmo **FORÇA-BRUTA**
  - Complexidade Assintótica: **O(NxM)**
  - Funcionamento:
    - Para cada possível posição inicial i (1 <= i <= N-M+1) o algoritmo testa se os M caracteres que começam a partir da i-ésima posição de T são idênticos à P.
    - Ex.: T = **GCCTGATGAACTTAG**, N = 15, P = **ACTTA**, M = 5



Qual é a **SOLUÇÃO?**

2. Para textos grandes

- A longa seqüência de caracteres pode ser armazenada em um grande vetor, **OU**,
- Ser **dividida em vários vetores menores**. (Técnica conhecida como **SBH - Sequencing by Hibridization**).
- **Principal desvantagem desta solução**: o grande volume de vetores necessários para esta divisão.
- **Principal vantagem desta solução :** as pesquisas nos vetores podem ser feitas ao mesmo tempo.

Como funciona o SBH

- Primeiro passo: Identificar todas as strings de tamanho 2K que são substrings de uma String S.

- Por que fazer isso?
  - Por exemplo, supondo que AC, CA e CC são as únicas substrings de tamanho 2 formadas a partir de uma string qualquer S.

  - Uma vez que a string S é formada pela concatenação das substrings acima, é fácil identificar que ACCA é uma substring de S. Já CAAC não é uma substring de S (pois tem a seqüência AA).

Como funciona o SBH

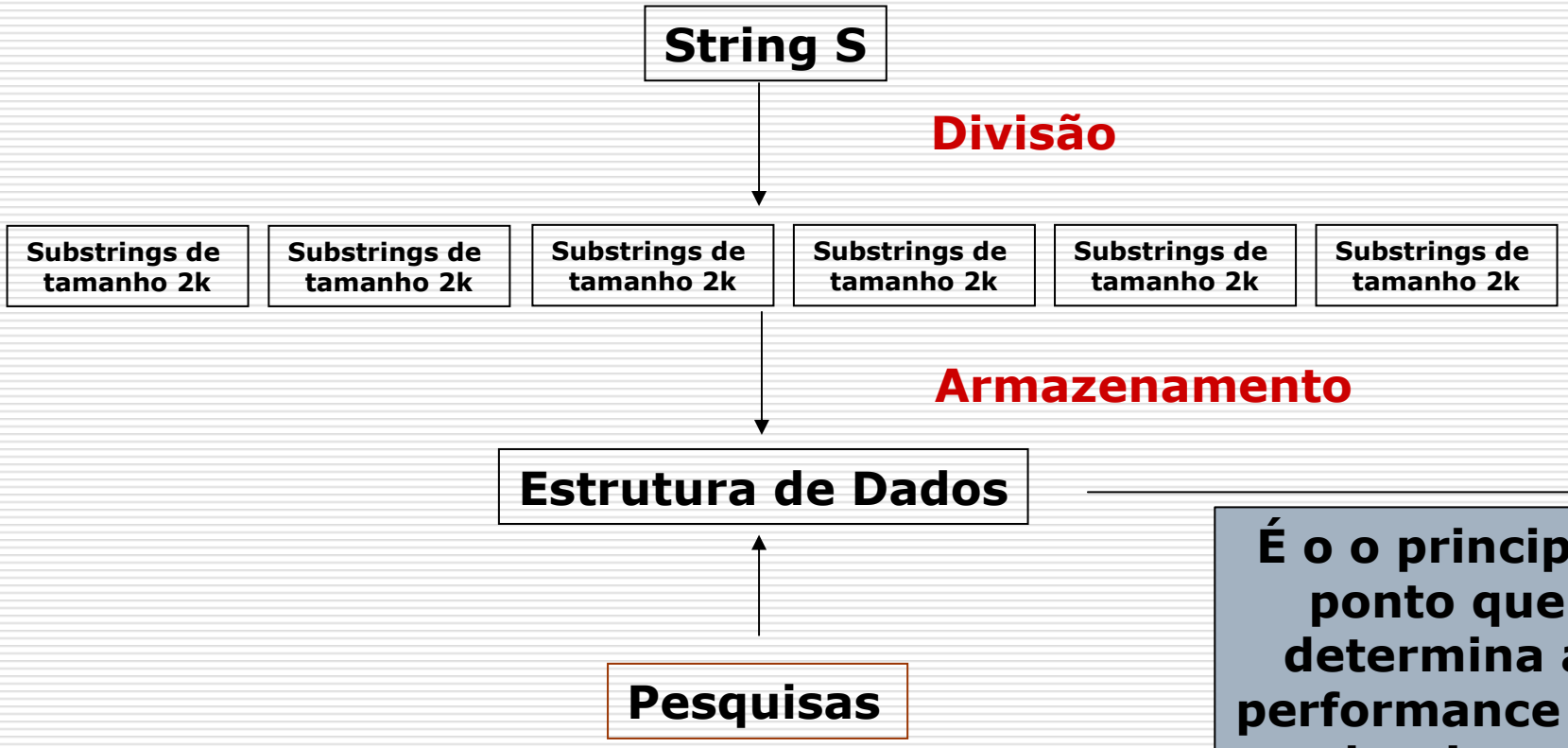
- O primeiro problema é encontrar um algoritmo *rápido* para construir todas as strings de tamanho 2K de S, considerando que S é muito longo.

- Uma vez que todas as substrings foram geradas, também necessita-se de uma estrutura de dados para armazená-las.

  - Esta estrutura de dados (**dicionário**) irá determinar o tempo que o algoritmo irá levar para descobrir se uma substring pertence ou não ao conjunto.

Como funciona o SBH

- **Resumindo:**



Formas de implementar o dicionário do SBH

- O autor testou quatro estruturas de dados
  - 1. Binary Tree – O(K Log n), onde k = nº de pares de caracteres e n = número de comparações.
  - 2. Hash Table – O(Log n)
  - 3. Suffix Tree
  - 4. Compressed Suffix Tree

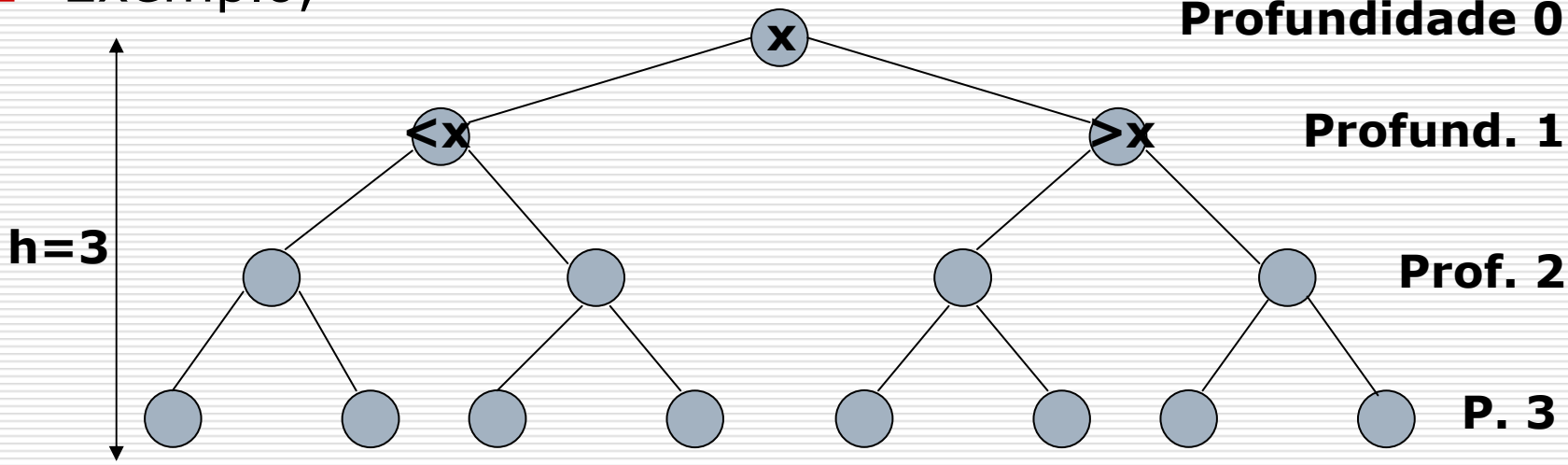
Comparação entre as Soluções

String Length	Binary Tree	Hash Table	Suffix Tree	Comp. Suffix
8	0,0	0,0	0,0	0,0
16	**0,0**	**0,0**	**0,0**	**0,0**
32	0,1	0,0	0,0	0,0
64	**0,3**	**0,4**	**0,3**	**0,0**
128	2,4	1,1	0,5	0,0
256	**17,1**	**9,4**	**3,8**	**0,2**
512	31,6	67,0	6,9	1,3
1.024	**1.828,9**	**96,6**	**31,5**	**2,7**
2.048	11.441,7	941,7	553,6	39,0
4.096	**> 2 dias**	**5.246,7**	**Out of Mem.**	**45,4**
8.192		> 2 dias		642,0
16.384				**1.614,0**
32.768				13.657,8
65.536				**39.776,9**

* Tempo em segundos

Binary Tree

- Árvore binária é uma árvore onde **cada nó contém no máximo dois filhos**;
- Cada filho é identificado como **filho da esquerda** ou **filho da direita**.
- Exemplo,



Binary Tree

- Algoritmo de busca de em árvore binária
  - Entradas: x = Posição na árvore, k = valor a ser encontrado
  - Saída: posição em que o valor está na árvore (x)

- BinaryTreeQuery(x,k)

```
if (x = nil) or (k=key[x])
        then return x
else if (k < key[x])
        then return BinaryTreeQuery(left[x],k)
        else return BinaryTreeQuery(right[x],k)
```

Complexidade: O(h)

Fim

- Outro integrante do grupo:
  - 2. Hash Table
  - 3. Suffix Tree
  - 4. Compressed Suffix Tree
- Obrigado!!!