"Strings of Pearls" - A Eficiência no Processamento de Textos

Por Felipe Stein e Arthur da Matta

10 de março de 2026

O Problema Central da Coluna 15

O problema central não é apenas "manipular texto", mas sim **como escolher a estrutura de dados correta para cada tipo de busca em grandes volumes de dados.** O autor mostra que, dependendo da sua necessidade (contar palavras, achar repetições ou gerar texto), uma estrutura simples pode ser lenta demais e uma complexa pode ser desnecessária.

Contagem e Frequência

O objetivo aqui é ler um documento e contar quantas vezes cada palavra aparece.

Abordagem STL (C++)

O autor começa usando `set` e `map` da biblioteca padrão. É elegante e funciona, mas pode ser lento para arquivos gigantescos como a Bíblia.

Aprimoramento com Hash Table

Para ganhar velocidade, ele implementa sua própria **Tabela Hash** em C.

Resultado

A implementação customizada em C chegou a ser **uma ordem de magnitude (10x) mais rápida** que as bibliotecas padrão, processando milhares de palavras em menos de um segundo.

Lição

Árvores de busca (usadas no STL) mantêm tudo ordenado, mas se você só quer contar, o Hashing é imbatível.

O Sufixo e a Repetição

Aqui o problema muda: **"Qual é a maior frase que se repete em um livro?"** (Ex: No livro *Ilíada*, de Homero).

1 O Problema da Força Bruta

Comparar todas as combinações de frases levaria um tempo absurdo ($O(n^2)$).

2 A Solução: Suffix Array (Vetor de Sufixos)

Esta é a **"pérola" do capítulo**. Em vez de copiar o texto, criamos um array de **ponteiros** onde cada ponteiro aponta para uma posição do texto original (cada posição é o início de um "sufixo").

3 A Técnica

Você ordena esses ponteiros alfabeticamente. Ao ordenar, frases repetidas ficam vizinhas uma da outra no array. Basta comparar os vizinhos para achar a maior repetição.

4 Eficiência

Isso transforma um problema impossível em algo que roda em segundos ($O(n \log n)$).

Geração de Texto e Cadeias de Markov

Como gerar um texto que pareça real (antecessor dos modelos de IA atuais)?

O Conceito

O autor usa **Cadeias de Markov**. A ideia é: dada uma sequência de *k* palavras, qual a probabilidade da próxima palavra aparecer?

Implementação

Ele usa a mesma estrutura de **Suffix Array** da parte anterior para encontrar rapidamente todas as ocorrências de uma frase e escolher a próxima palavra aleatoriamente entre as opções reais do texto original.

Níveis de Ordem

01

Ordem-1

Escolhe a próxima letra/palavra baseada apenas na anterior (gera um texto sem sentido).

02

Ordem-2 ou 3

2:Ele olha as duas últimas palavras para decidir a terceira.

3:Ele olha as três últimas para decidir a quarta.