Algoritmo de Huffman

Estrutura de dados

Orleoncio Maciel de Oliveira Filho
Joao Victor Pequeno Damasceno
Thales Lucas Lima e Gomes
Yan Pedro Façanha Brasileiro

Compressão Sem Perdas

Definição e Princípios:

- Lossless Compression reduz o tamanho de arquivos sem perder nenhuma informação (preserva todos os dados originais).
- Baseada em identificação e eliminação de redundâncias nos dados.
- Uso fundamental onde a integridade dos dados é crucial (transmissão de informações sensíveis, armazenamento de arquivos importantes).

Desafios:

- Encontrar um equilíbrio entre a taxa de compressão e a velocidade de compactação/descompactação.

Como Funciona?

Funcionamento:

- A compressão sem perdas identifica padrões repetitivos nos dados e os substitui por representações mais compactas.
- Objetivo: Reduzir o tamanho do arquivo sem comprometer a integridade dos dados.

Papel do Huffman:

- O Algoritmo de Huffman é uma das técnicas de compressão sem perda de dados.
- Ele é usado para criar uma representação compacta dos dados, atribuindo códigos binários menores aos símbolos mais frequentes.

Construção da Árvore

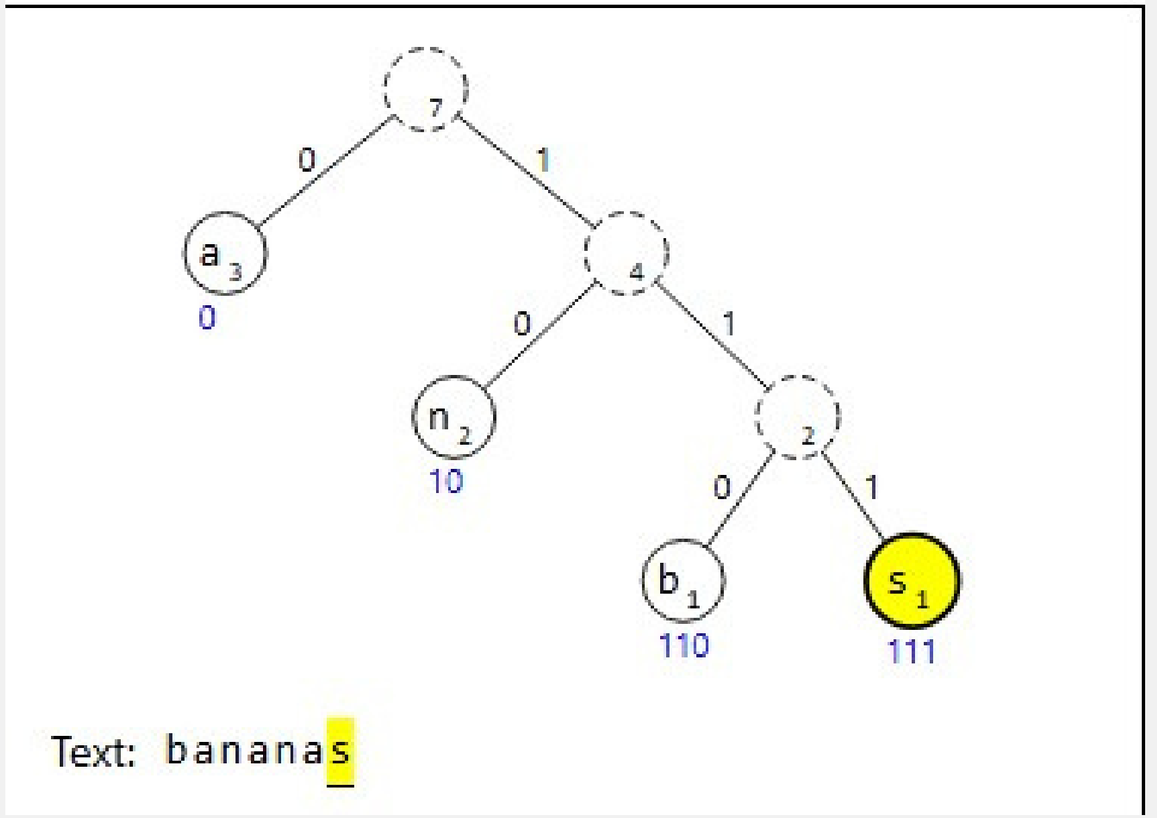
Ideia central: Associar números binários com menos bits aos caracteres mais usados em um texto.

Dividida em 3 fase: Cálculo da frequência de cada caractere → Construção da árvore binária (Árvore de Huffman) → Codificação propriamente dita.

A árvore é binária e baseada na frequência de uso das letras, sendo construída das folhas para a raiz (de baixo para cima).

Passo-a-passo:

1. Inicialmente, cada letra é uma árvore contendo apenas sua frequência (raiz).
2. Escolhem-se as duas árvores com as menores frequências.
3. Elas são transformadas em uma única árvore cujo valor é a soma das duas.
4. O processo repete até restar apenas uma única árvore.



Codificação Prefix-free

Definição e Princípios:

- Nenhuma palavra-código é prefixo de outra, garantindo decodificação inequívoca.
- Exemplo: Se a=0, nenhum outro código pode começar com 0.
- Códigos de Huffman são de comprimento variável, usando códigos menores para símbolos mais frequentes.

Desafios:

- É possível ter um código de comprimento variável com prefixo, mas a codificação e a decodificação são sempre mais complexas.

Uso de heap

- Utiliza-se uma fila de prioridade mínima, geralmente um min-heap.
- Função: Gerenciar e recuperar rapidamente os nós com menores frequências.

Processo:

- O heap armazena símbolos e frequências.
- Extração dos dois menores valores: O (log n).
- Inserção do novo nó somado: O (log n).

- A complexidade total é O (n log n). Sem heap, a complexidade seria (n^2), impraticável para grandes dados.

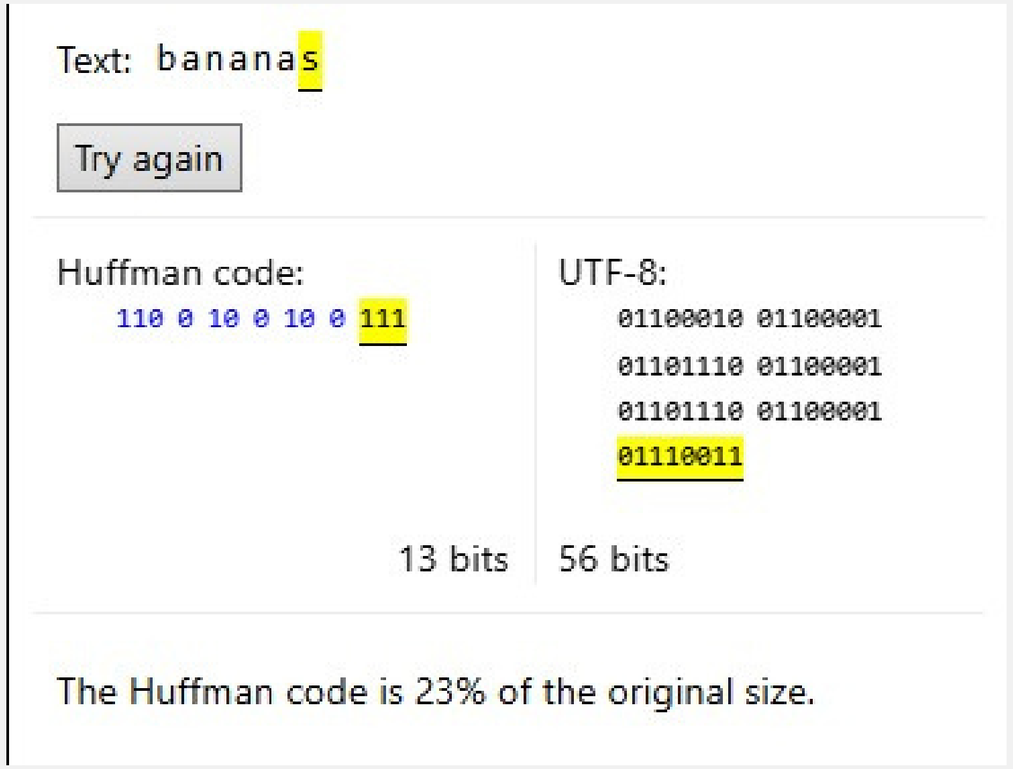
Codificação e Decodificação

Codificação: Aresta para filho esquerdo = 0, aresta para filho direito = 1; O código é o caminho da raiz até a folha.

É possível criar uma tabela de pesquisa (lookup table), para agilizar a codificação, mapeando símbolo à código binário.

A árvore é binária e baseada na frequência de uso das letras, sendo construída das folhas para a raiz (de baixo para cima).

Decodificação: Lê-se bit a bit percorrendo a árvore da raiz → Se 0, vai para esquerda; se 1, vai para direita → Ao atingir uma folha, o caractere é identificado e o processo reinicia na raiz.



Letter	Huffman Code
a	0
n	10
b	110
s	111

Aplicações: ZIP, MP3 e PNG

ZIP: Substitui sequências repetidas por referências (ex: "volte 6 bytes, copie 5") → Aplica Huffman nos literais e nas referências de distância/tamanho.

MP3: Remove frequências mascaradas ou imperceptíveis ao ouvido humano (onde ocorre a maior perda) → Quantiza (arredonda) os valores restantes agressivamente → Huffman entra no final para codificar eficientemente os valores quantizados que sobraram.

PNG: Aplica filtros para tornar os dados dos pixels mais previsíveis → Usa Deflate (LZ77 + Huffman) nos dados filtrados.