BST and Heap : Huffman coding and decoding

Cristiano Lima Sousa Rosa
UFRR - Universidade Federal de Roraima
Boa vista - RR - Brasil
cristiano.16bits@gmail.com

Abstract - This report in article format, looking for how the Huffman algorithm works, is used for data compression. Analyze its complexity and its applications.

Resumo - Este relatório em formato de artigo busca entender o funcionamento do algoritmo Huffman, que é utilizado para compressão de dados. Analisar sua complexidade e suas aplicações.

I - Introdução

O algoritmo de codificação huffman foi desenvolvido em 1952 por David A. Huffman, método de compressão de dados, sendo bastante útil. É criada a codificação a partir da tabela ou vetor de recorrência de letras para que seja adicionado na fila de prioridade, foi utilizada como entrada um arquivo em HTML para ter várias recorrências.

II - Codificação Huffman

O algoritmo huffman realiza uma codificação usando árvores binárias e fila de prioridade mínima, para construção do algoritmo. A figura 1 mostra uma tabela contendo a letra e sua respectiva recorrência que será utilizada pelo huffman, Figura 2 representa a fila de prioridade e a figura 3 representa o novo Nó sendo formado pelos 2 Nós de menor prioridade da fila, sua prioridade será a soma das prioridades dos Nós retirados.

Letra	Frequência
A	3
B	1
C	1

Figura 1. Tabela de recorrência.

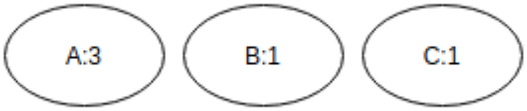

Figura 2. Representação da fila de prioridades.

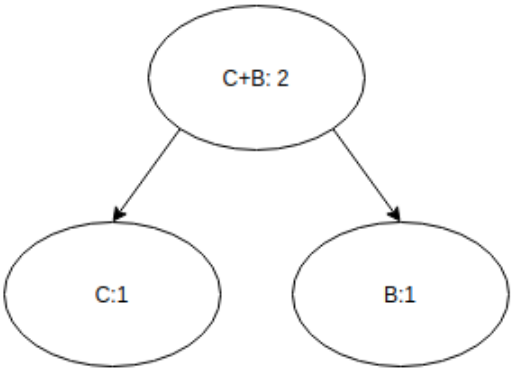

Figura 3. Árvore sendo construída a partir do novo Nó.

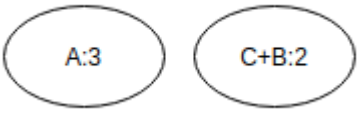

Figura 4. Nova fila de prioridade formada a partir da nova inserção.

Com os dois últimos Nós será novamente retirado e criador o novo Nó com a junção das duas, retornando a árvore para codificação e decodificação.

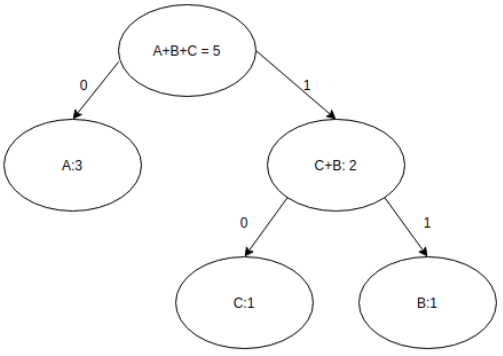

Figura 5. Árvore final retornada do huffman.

Letra	Codificação
A	0
B	11
C	10

Figura 6. Tabela para codificação.

C	C	B	B	CCBB
10	10	11	11	10101111

Figura 7. Codificação.

O custo de representar um caractere na tabela ASCII é de 8 bits, portanto o cálculo para saber a quantidade de armazenamento da sequência basta multiplicar a quantidade de letras por 8. A sequência CCBB usando ASCII é 4x8 = 32 bits, usando o algoritmo huffman são utilizados 8 bits observada na figura 7.

III. implementação

Os algoritmos foram desenvolvidos na linguagem de programação C e com base nos pseudocódigos do livro Algoritmos: *Teoria e Prática. 3a edição[1].* Para a elaboração do projeto foi necessário criar primeiramente uma forma de ler os caracteres do arquivo em HTML, A linguagem C nos disponibiliza funções para leitura de arquivos passando por cada caractere até o EOF ou fim de arquivo, Como a tabela ASCII podemos ter a relação do número em decimal com um tipo Char, isto é , o número 65 quando transformado em Char será a letra A. ASCII tem um total de 255 caracteres, logo podemos ter um vetor de 255 posições e aproveitar seu número como índice no vetor e ao percorrer o arquivo fazemos o inverso, pegamos a letra e transformamos em índice numérico para incrementar sua frequência na própria posição do vetor, tendo assim um vetor de frequência.

Índice do array	65	66	67
Representação em Char	A	B	C
Frequência	0	0	3

Figura 8. Demonstração do armazenamento da recorrência.

A criação da fila de prioridade mínima é criada por n elementos que tenham frequência diferente de 0 sendo necessário percorrendo o vetor tendo um custo O(n) onde n é às 255 posições , a construção de node ou nós que contém o seu caractere e frequência, pela heap mínima irão ser adicionados na fila e nos locais devidos, o menor valor estará na primeira posição contendo acesso de O(1).

Com a fila definida podemos iniciar o algoritmo huffman construindo um novo nó que irá receber 2 nós retirados da fila de prioridade mínima que serão adicionados no lado esquerdo e direito do novo nó, sua prioridade é definida pela soma dos dois nós retirados depois disso o nó será adicionado na fila. Quando o algoritmo estiver na reta final restará apenas um nó que conterá uma árvore que servirá para a codificação huffman. Para obter a tabela de codificação temos que vasculhar a árvore gerada até encontrar folhas da árvore exemplificadas na figura 5, armazenar as direções escolhidas sendo 0 para esquerda e 1 para direita.

IV - Análise de complexidade

O Pseudocódigo a seguir foi retirado do livro Algoritmos: Teoria e Prática[1]

```
HUFFMAN(C) [1]
1 ) n <− | C |
2 ) Q <− C
3 ) for  i <− 1 to n−1
4 )        do alocar um novo nó z
5 )        esquerda[ z ] <− x <−EXTRACT−MIN(Q)
6 )        direita [ z ] <− y <− EXTRACT−MIN (Q)
7 )        f [ z ] <− f [ x ] + f [ y ]
8 )        INSERT (Q, z )
9 ) r e t u r n EXTRACT−MIN (Q)
```

n é a quantidade de caracteres presentes na fila e Q sendo a min-heap ou fila de prioridade mínima.
Como as operações na heap tem custo O(logn) pois o tempo é proporcional a altura da árvore e o loop na linha 3 e 8 são n-1 logo o custo do algoritmo é O(nlogn).

V - Conclusão

A implementação do algoritmo huffman nos dá entendimento do funcionamento da forma de compressão de dados, o exemplo de tabela gerada representada na figura 6 contendo a sequência de bits necessária para representar um caractere ou sequência de letras, como na figura 7, é uma redução bastante vantajosa comparada com a representação dos caracteres na tabela ASCII que

precisam de 8 bits para representar apenas um caractere. Podemos aplicar a codificação nos dias atuais, seja na forma compactação e/ou descompactação de mensagens ou de imagens gerada pela codificação huffman. Infelizmente não consegui fazer a decodificação apenas codificação e salvar a tabela de codificação num arquivo .txt . O código desenvolvido está presente no github abaixo

Link do repositório:
https://github.com/K16bits/Cristiano_FinalProject_AA_RR_2021.git

Execução:

./huffman nomedoarquivo.html

Será criado um arquivo registro.txt com as letras e suas codificações.

Saída:

no terminal apresentará as letras do html codificado.

REFERÊNCIAS

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. **Algoritmos: Teoria e Prática. 3a edição**. Elsevier, 2012.
2. Paulo Feofiloff, **Algoritmo de Huffman para compressão de dados** Disponivel em <https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/huffman.html>