Flávio Eduardo Oliveira e Silva – 2017018013

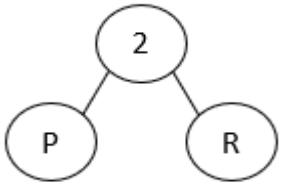
3 - Código de Huffman:

1º Passo: Ordenar em ordem crescente de frequência:

P	R	F	V	S	N	H	D	M	I	O	E	A
1	1	2	2	2	3	4	4	6	6	8	10	15

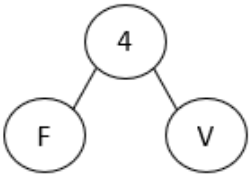
2º Passo: Juntar dois caracteres de menor frequência:

P+R	F	V	S	N	H	D	M	I	O	E	A
2	2	2	2	3	4	4	6	6	8	10	15



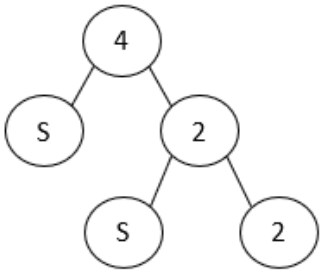
3º Passo: Juntar outros dois caracteres de menor frequência:

P+R	S	N	F+V	H	D	M	I	O	E	A
2	2	3	4	4	4	6	6	8	10	15



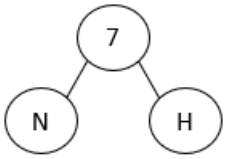
4º Passo: Juntar um caracter com um conjunto de caracteres de menor frequência:

N	P+R+S	F+V	H	D	M	I	O	E	A
3	4	4	4	4	6	6	8	10	15



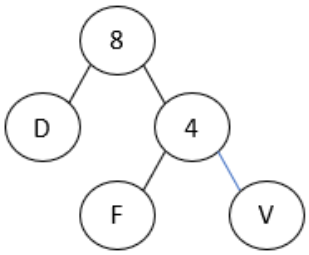
5º Passo: Juntar outros dois caracteres de menor frequência:

P+R+S	F+V	D	M	I	N+H	O	E	A
4	4	4	6	6	7	8	10	15



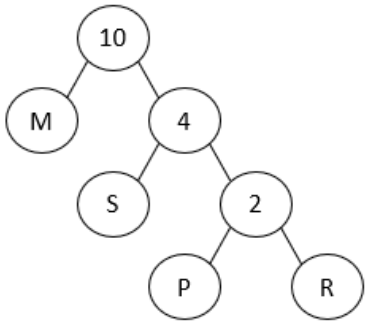
6º Passo: Juntar um caracter com um conjunto de caracteres de menor frequência:

P+R+S	M	I	N+H	F+V+D	O	E	A
4	6	6	7	8	8	10	15



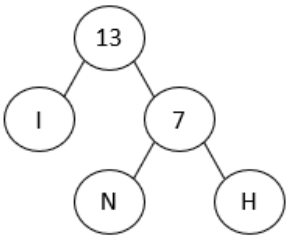
7º Passo: Juntar um caracter com um conjunto de caracteres de menor frequência:

I	N+H	F+V+D	O	P+R+S+M	E	A
6	7	8	8	10	10	15



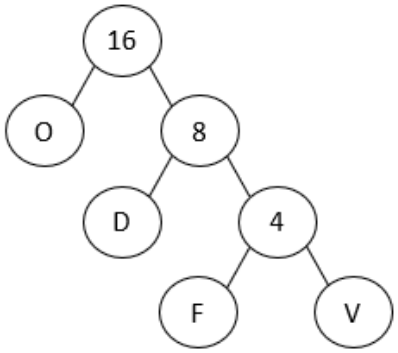
8º Passo: Juntar um caracter com um conjunto de caracteres de menor frequência:

F+V+D	O	P+R+S+M	E	N+H+I	A
8	8	10	10	13	15



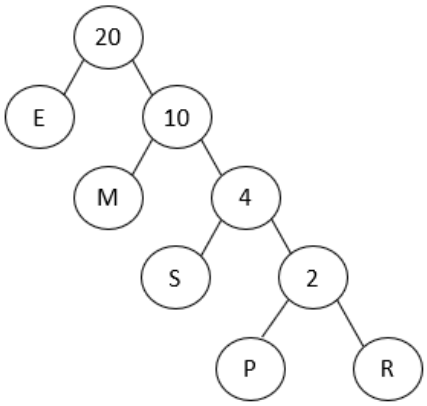
9º Passo: Juntar um caracter com um conjunto de caracteres de menor frequência:

P+R+S+M	E	N+H+I	A	F+V+D+O
10	**10**	**13**	**15**	**16**



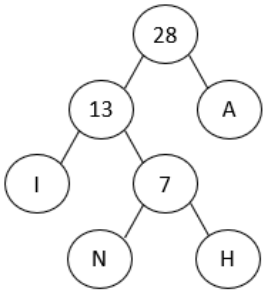
10º Passo: Juntar um caracter com um conjunto de caracteres de menor frequência:

N+H+I	A	F+V+D+O	P+R+S+M+E
13	**15**	**16**	**20**



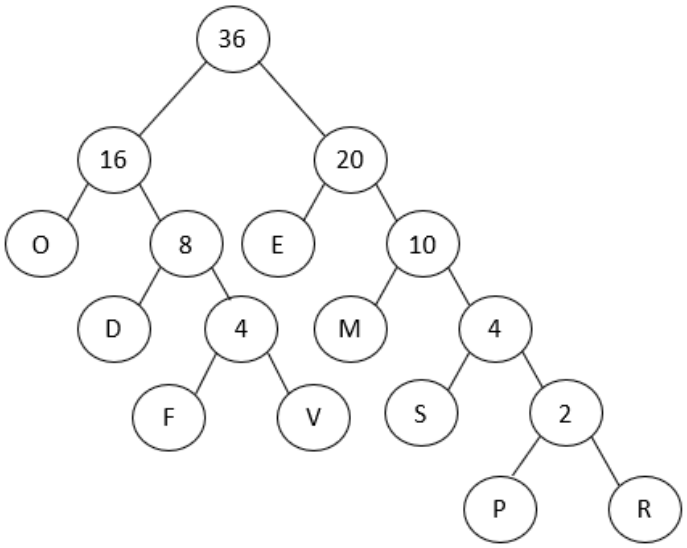
11º Passo: Juntar um caracter com um conjunto de caracteres de menor frequência:

F+V+D+O	P+R+S+M+E	N+H+I+A
16	**20**	**28**



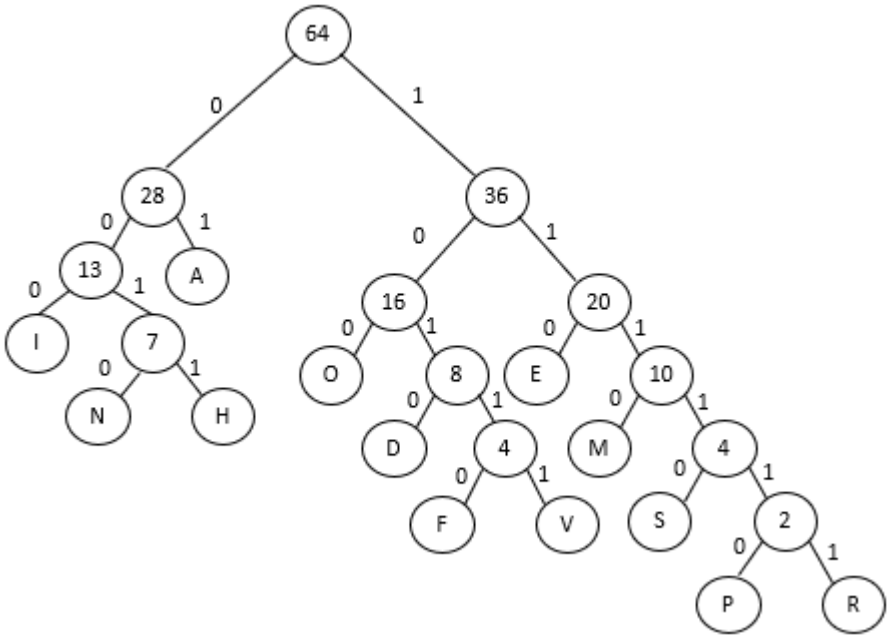
12º Passo: Juntar um caracter com um conjunto de caracteres de menor frequência:

N+H+I+A	P+R+S+M+E+F+V+D+O
28	36



13º Passo: Juntar um caracter com um conjunto de caracteres de menor frequência:

P+R+S+M+E+F+V+D+O+N+H+I+A
64



A árvore de Huffman está gerada, com ela é possivel codificar a mensagem “aprovafoiadiada”:

P	R	F	V	S	N	H	D	M	I	O	E	A
111110	**111111**	**10110**	**10111**	**11110**	**10**	**0011**	**1010**	**1110**	**000**	**100**	**110**	**01**

Codificação da mensagem:

01111110111111100101110110110100000011010000011010101

4 – Algoritmo:

Min_Env(p,n)

1. num_min_envolp <- 0,j <- n
2. MergeSort(p,n)
3. para i <- 1 até i<= j faça
4. se (p[i] + p[j] <=1)
5. entao i<- i+1
6. j <- j-1
7. num_min_envolp <- num_min_envolp + 1
8. devolve num_min_envolp

Consumo de tempo: O(nlgn)

Linhas	Consumo
1	O(1)
2	O(ngln)
3	O(n)
4	O(n)
5	0
6	O(n)
7	O(n)
8	O(1)

Considerando o pior caso que todos os envelopes conterão apenas um livro, todos os livros conterão pesos muito altos, não tornando possível colocá-los junto com outros dentro dos envelopes, portanto, a linha 5 nunca será executada.

Cálculo**: 2.O(1) + O(nlgn) + 4.O(n) + 0 = O(nlgn)**

A linha 3 garante que todos os pesos dos livros serão verificados e as linhas 5 e 6 garantem que o algoritmo tem parada.

A linha 4 sempre verificará se é possível colocar o livro de peso[i] com o livro de peso[j] no mesmo envelope, de forma que não esceda o peso 1 e obtenha-se o número mínimo de envelopes. A linha 7 contabilizará o número mínimo de envelopes que serão precisos o qual é retornado pela linha 8.

O algoritmo é do tipo guloso, pois ele sempre vai colocar no envelope o livro de pesop[j] junto ao livro de peso[i] caso a soma de seus pesos for menor ou igual a 1. Caso contrário, o livro de p[j] ficará sozinho no envelope e a execução continua comparando-se o peso de p[i] com p[j-1]. Logo, no fim da execução, ele retorna o menor número possível de envelopes.