UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Código de Huffman

Disciplina: Algoritmos 2

Alunos: Juliana Cavalcante,
Maximiliano Alves

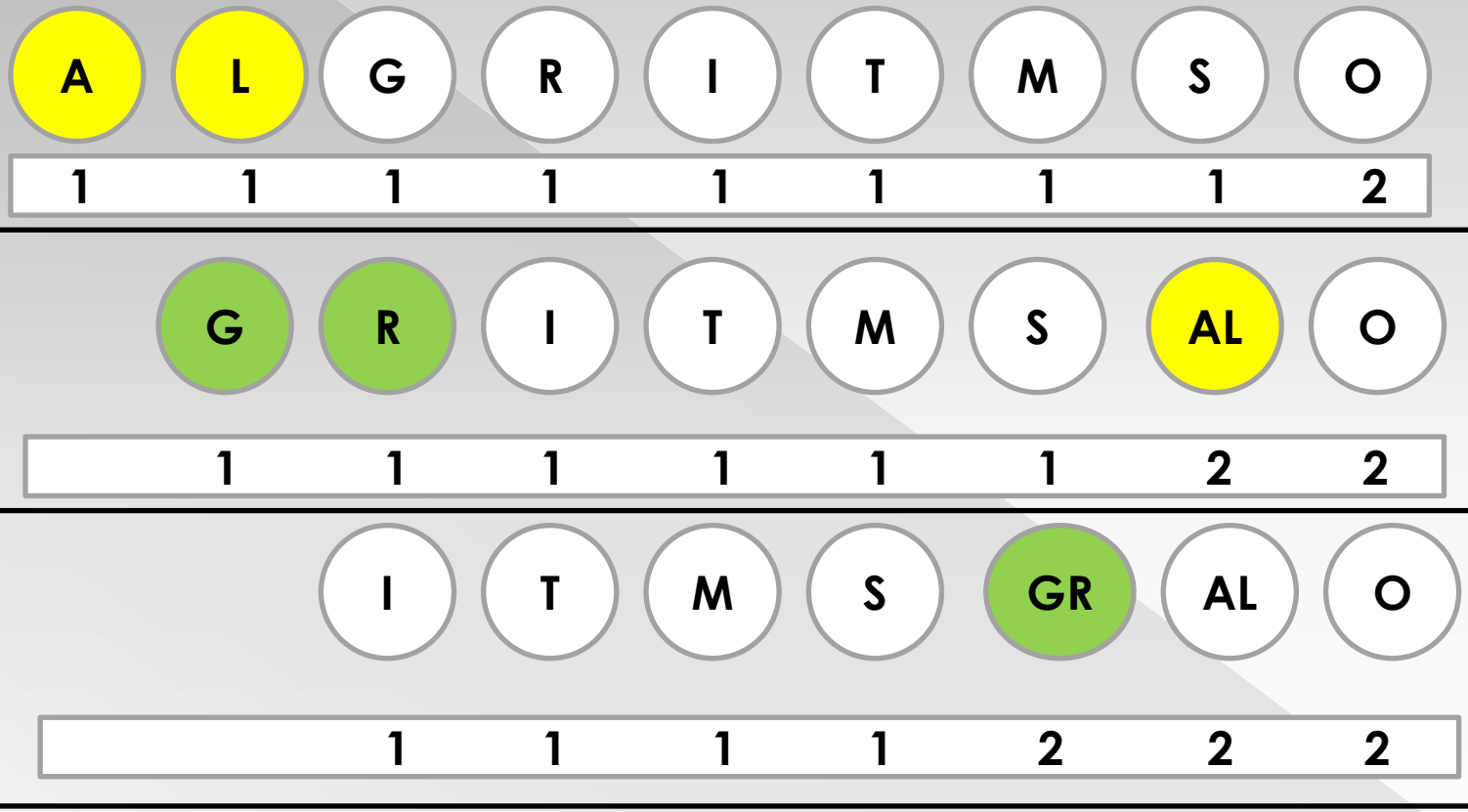
Definição

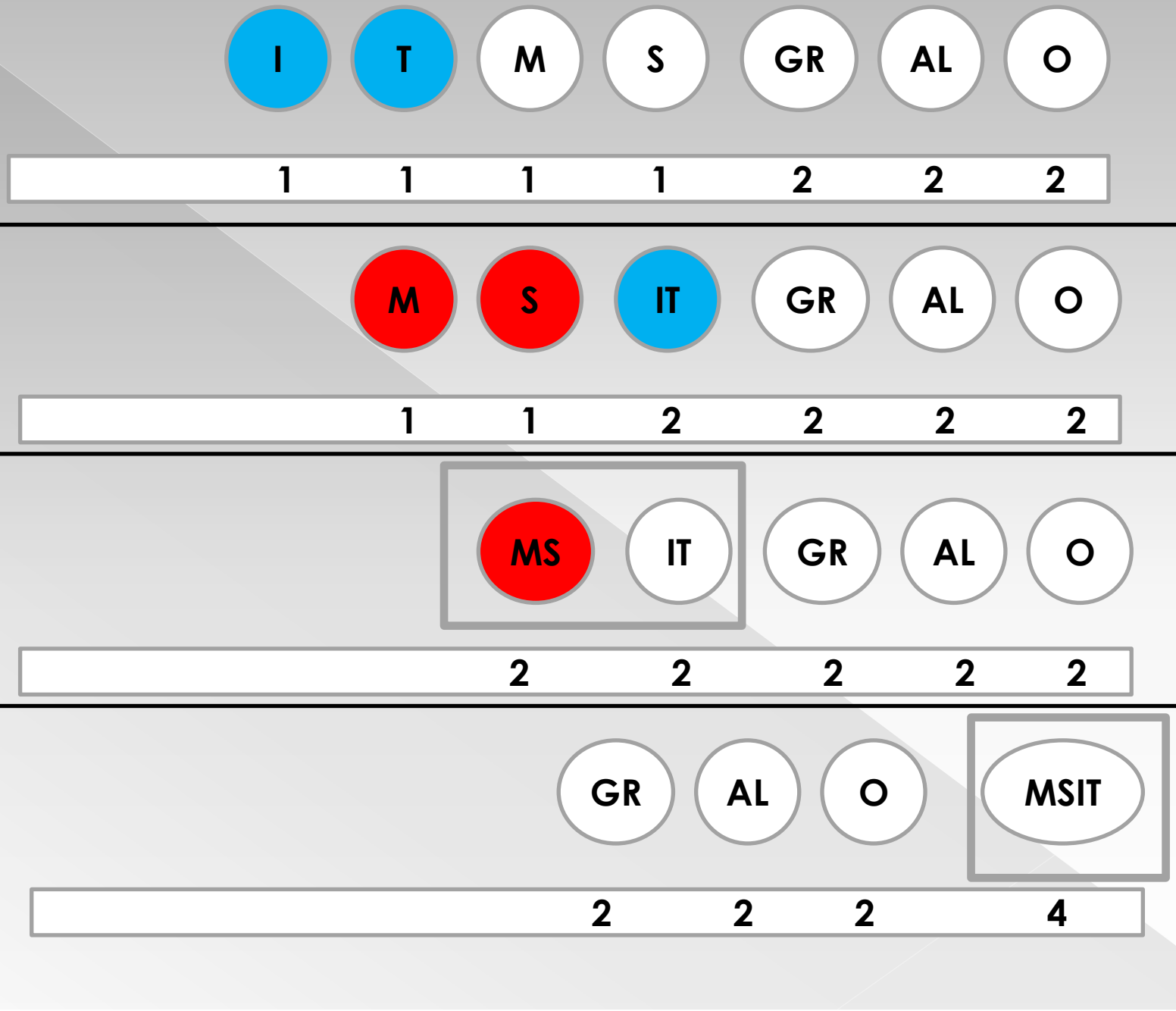
- A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman.

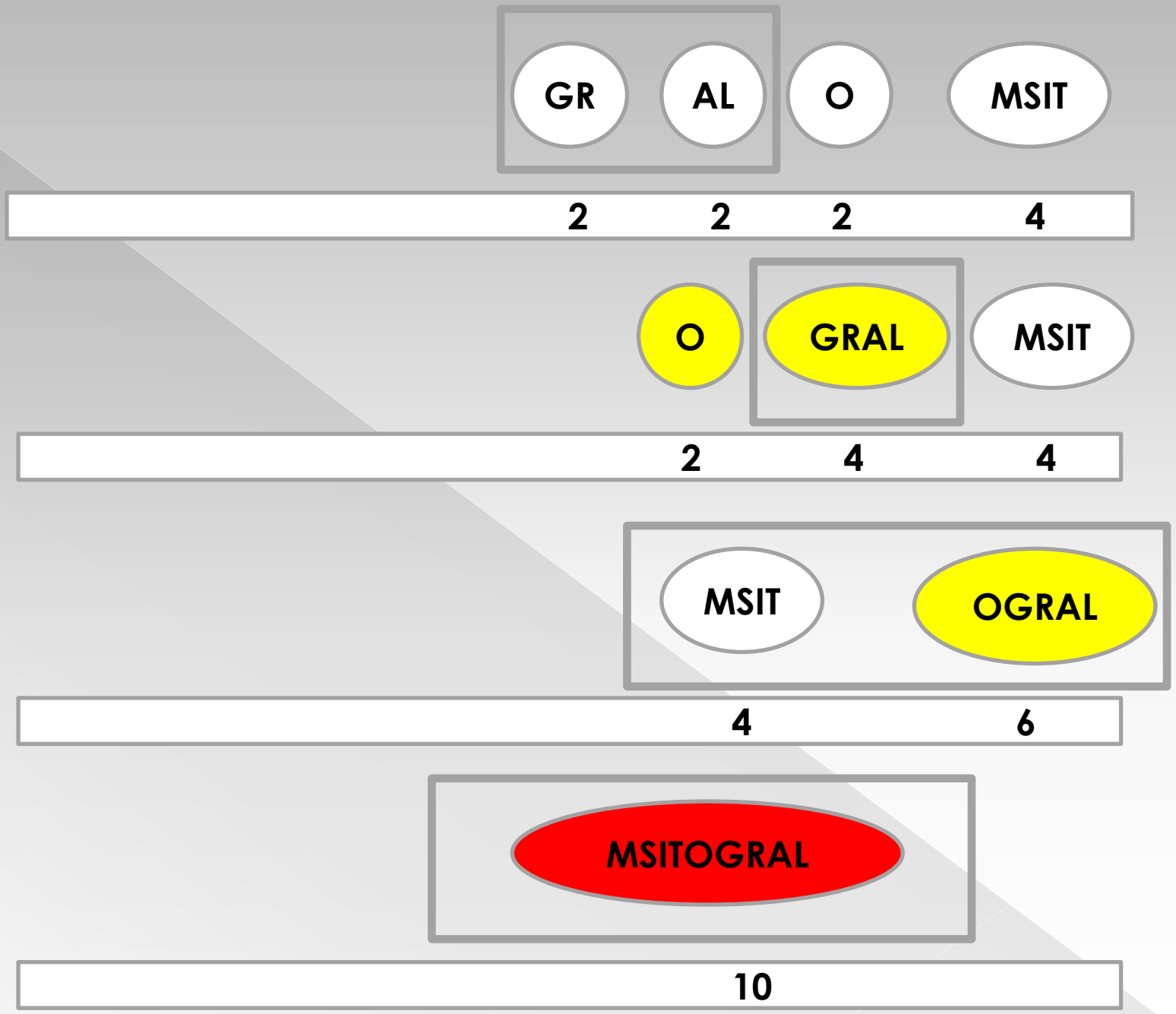
Funcionamento

- Uma árvore binária completa, chamada de árvore de Huffman é construída recursivamente a partir da junção dos dois símbolos de menor probabilidade, que são então somados em símbolos auxiliares e estes símbolos auxiliares recolocados no conjunto de símbolos.

Exemplo: "Algoritmos"




I
T
M
S
GR
AL
O
1 1 1 1 2 2 2
M
S
IT
GR
AL
O
1 1 2 2 2 2
MS
IT
GR
AL
O
2 2 2 2 2
GR
AL
O
MSIT
2 2 2 4


GR
AL
O
MSIT
2
2
2
4
O
GRAL
MSIT
2
4
4
MSIT
OGRAL
4
6
MSITOGRAL
10

O pseudocódigo

```
enquanto tamanho(alfabeto) > 1:
  S0 := retira_menor_probabilidade(alfabeto)
  S1 := retira_menor_probabilidade(alfabeto)
  X  := novo_nó
  X.filho0 := S0
  X.filho1 := S1
  X.probabilidade := S0.probabilidade + S1.probabilidade
  insere(alfabeto, X)
fim enquanto

X = retira_menor_símbolo(alfabeto) # nesse ponto só existe um símbolo.

para cada folha em folhas(X):
  código[folha] := percorre_da_raiz_até_a_folha(folha)
fim para
```

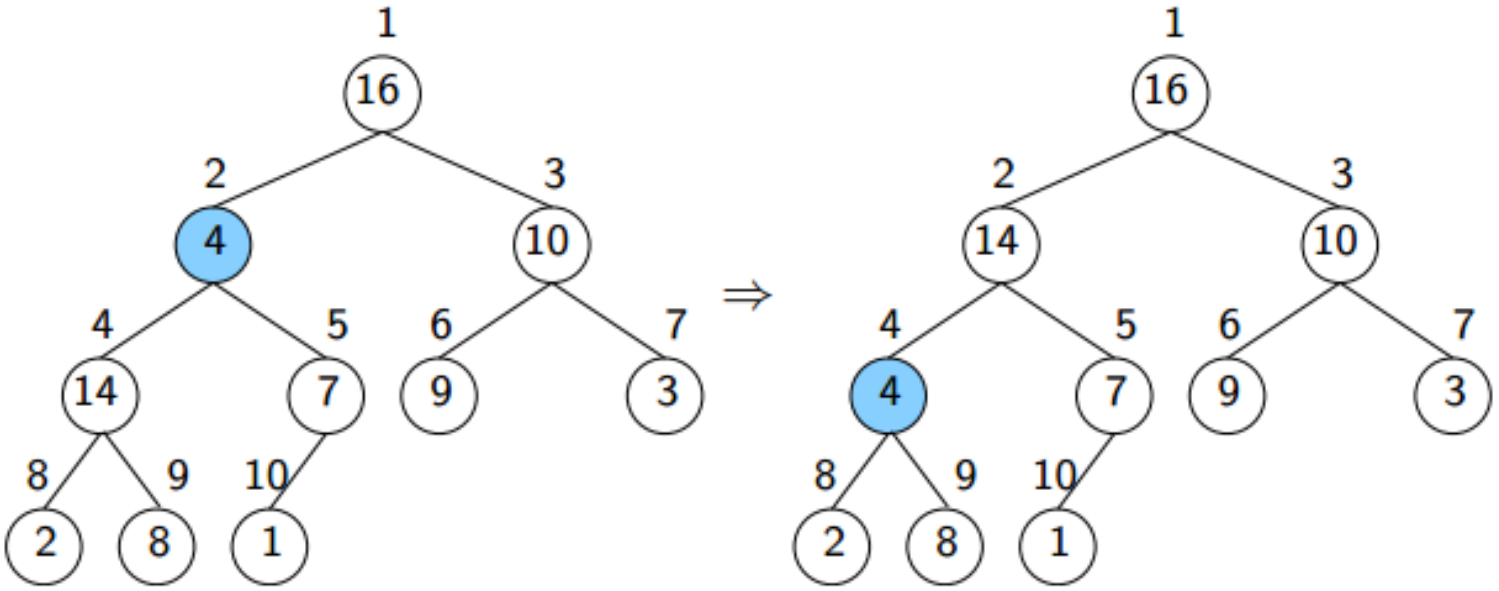
Heap

- Um heap (binário) é uma estrutura de dados que pode ser vista como uma árvore binária completa.
- Pode funcionar como uma fila de prioridades.

- Cada nó da árvore corresponde a um elemento do vetor que armazena o valor do nó. A árvore é completa em todos os níveis com exceção do nível mais baixo, o qual é completado da esquerda para a direita.

- Para que um vetor A seja heap, cada nó i, com exceção da raiz, deve satisfazer a seguinte propriedade: o valor de cada nó é no máximo o valor do seu pai.

Exemplo de heap



Elaboração do código de Huffman

```c
typedef struct No{
    char c;
    int freq;
    struct No *esq, *dir;
}No;

typedef struct Arvore
{
    int tam;
    int capac;
    struct No **array;
}Arvore;

No* criaNo(char c, int freq);
Arvore* criaArvore(int capac);
void swapNo(No** a,  No** b);
void MinHeapify(Arvore* arvore, int id);
void imprimeArray(int arr[], int n);
No* constroiArvHuffman(char c[], int freq[], int tam);
void imprimeCod(No* no, int arr[], int top);
No* achaMin(Arvore* arvore);
void insereArvore(Arvore* arvore,  No* no);
void constroiArvore(Arvore* arvore);
void codigoHuffman(char c[], int freq[], int tam);
int contaCaracteres(char c, char* string, int op);
```

```c
//verificando a propriedade de heap
void MinHeapify(Arvore* arvore, int id)
{
    int menor = id;
    int esq = 2 * id + 1;
    int dir = 2 * id + 2;

    if ( esq < arvore->tam && arvore->array[esq]->freq < arvore->array[menor]->freq){
    //se a frequencia dos nós da esquerda forem menores que a frquencia dos menores nós
      menor = esq;//então a esquerda é agora o menor
    }
    if (dir < arvore->tam && arvore->array[dir]->freq < arvore->array[menor]->freq){
      //se a frequencia dos nós da direita forem menores que a frquencia dos menores nós
      menor = dir;//então a direita é agora a menor
    }
    if (menor != id)//se o menor for diferente do id que veio da função
    {
        swapNo(&arvore->array[menor], &arvore->array[id]);//então troca os nós menor e id
        MinHeapify(arvore, menor);//chamada recursiva, até todos os nós estarem nas posições certas
    }
}
```

```
No* constroiArvHuffman(char c[], int freq[], int tam)
{
    No *esq, *dir, *topo;
    Arvore* arvore = criaArvore(tam);//cria a arvore de acordo com a capacidade especificada
    int i;
    for (i = 0; i < tam; ++i)
        arvore->array[i] = criaNo(c[i], freq[i]); //o array da arvore recebe os nos que estão sendo criados
    arvore->tam = tam;//seta o tamanho da arvore que foi criada
    constroiArvore(arvore);//chama a função que constroi a arvore
    while (arvore->tam != 1)//enquando o tamanho da arvore for diferente de 1
    {
        esq = achaMin(arvore);//extrai o minimo do nó esquerdo
        dir = achaMin(arvore);//extrai o minimo do nó direito
        topo = criaNo('  ', (int)(esq->freq+dir->freq));//cria um nó vazio com a soma das frequencias dos nós
        topo->esq = esq; // a esquerda do nó vazio é o minimo encontrado a esquerda
        topo->dir = dir; //a direita do nó vazio é o minimo encontrado a direita
        insereArvore(arvore, topo);
    }
    return achaMin(arvore);//retorna o valor minimo da arvora apos todo o processo
}
```

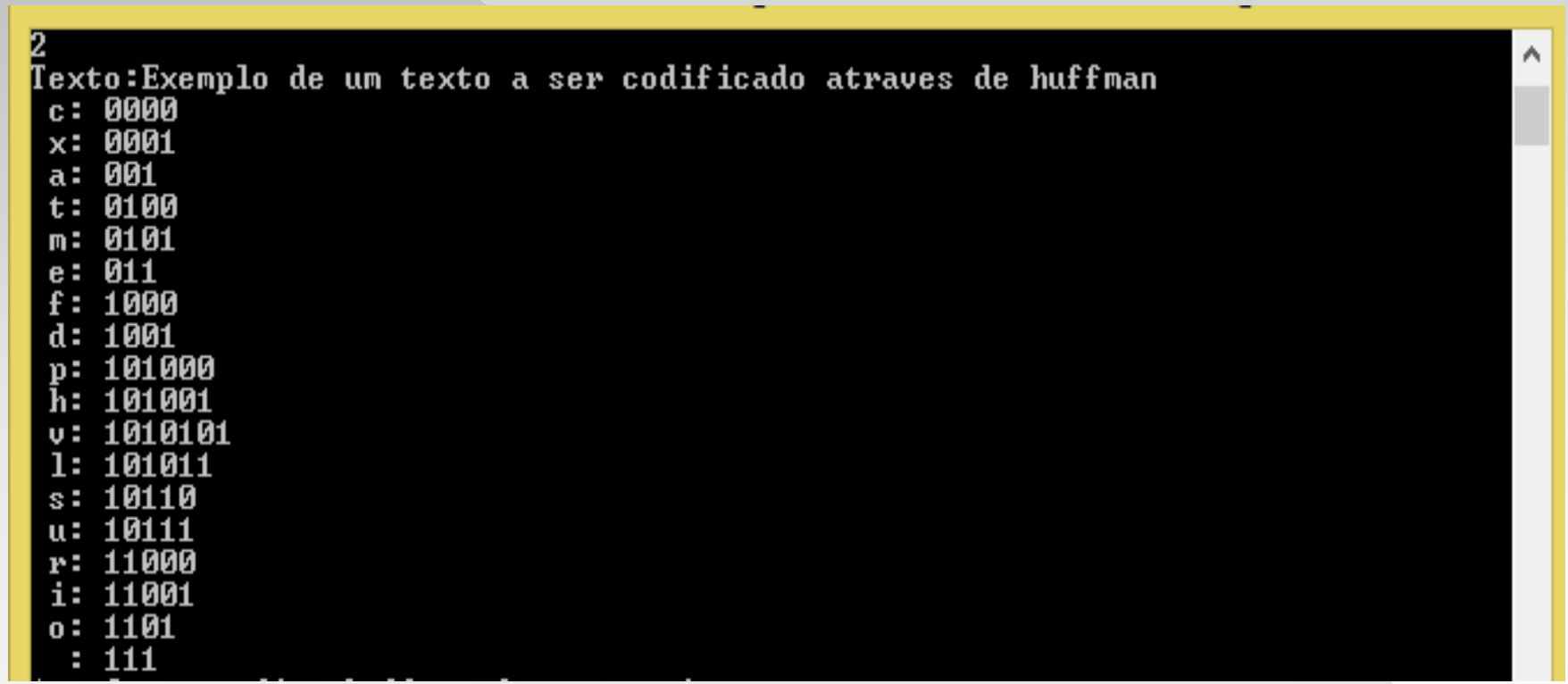
```
void imprimeCod(No* no, int vet[], int topo)
{//imprime na tela os codigos que são gerados ao percorrer a arvore criada
    if (no->esq!=NULL)//se a esquerda não for nula
    {
        vet[topo] = 0;//para o lado esquerdo o codigo referente é zero
        imprimeCod(no->esq, vet, topo + 1);//imprime o lado esquerdo
    }
    if (no->dir!=NULL)//se a direita não for nula
    {
        vet[topo] = 1;//para o lado direito o codigo referente é um
        imprimeCod(no->dir, vet, topo + 1);//imprime o lado direito
    }
    if (no->esq == NULL && no->dir == NULL)//se a direita e a esquerda forem nulas, o nó é uma folha
    {
        if(no->freq!=0){//não imprime as folhas onde a frequencia é nula para evitar processos desnecessarios
            printf(" %c: ", no->c);//imprime na tela o caracter
            imprimeArray(vet, topo);//chama a funcao que imprime o codigo de huffman gerado para ela
        }
    }
}
```

```
void codigoHuffman(char c[], int freq[], int tam)
{
    No* no = constroiArvHuffman(c, freq, tam);
    int vetor[MAX];
    imprimeCod(no, vetor, 0);//imprime na tela o codigo de huffman gerado
}
```

Resultados


1 - Gerar codigo huffman de um arquivo .txt
2 - Digitar um texto para gerar um codigo de huffman
3 - Sair
Digite a opcao:
2
Texto:Exemplo de um texto a ser codificado atraves de huffman

Resultados


2
Texto:Exemplo de um texto a ser codificado atraves de huffman
c: 0000
x: 0001
a: 001
t: 0100
m: 0101
e: 011
f: 1000
d: 1001
p: 101000
h: 101001
v: 1010101
l: 101011
s: 10110
u: 10111
r: 11000
i: 11001
o: 1101
: 111