**Laboratorio 4 - Huffman**

Estructuras de Datos Avanzadas

Universidad Nacional de San Agustín

Nombre: Alexander Rusvell Apaza Torres

1. Código Fuente:
   1. Clase hNodo: Esta clase será el nodo del árbol de HUffman que obtendremos.

class hNodo

{

private:

char letra;

int frecuencia;

hNodo\* izq;

hNodo\* der;

hNodo\* padre;

bool hoja;

string camino;

int id;

public:

hNodo(char letra, int frecuencia);

hNodo(int frecuencia);

string getStrValue();

~hNodo();

bool operator < (const hNodo& a){

if (this->frecuencia < a.frecuencia) return true;

return false;

}

friend ostream& operator<< (ostream& os, const hNodo& obj);

friend bool operator> (const hNodo& a, const hNodo& b);

friend hNodo\* operator + ( hNodo&, hNodo&);

friend class Huffman;

};

bool operator> (const hNodo& a, const hNodo& b){

return a.frecuencia > b.frecuencia;

}

//Constructor para nodos hoja

hNodo::hNodo(char letra, int frecuencia)

{

this->letra = letra;

this->frecuencia = frecuencia;

this->izq = 0;

this->der = 0;

this->padre = 0;

this->hoja = true;

}

//Constructor para nodos padre

hNodo::hNodo( int frecuencia)

{

this->frecuencia = frecuencia;

this->izq = 0;

this->der = 0;

this->padre = 0;

this->hoja = false;

}

* 1. Clase Huffman: Esta clase genera el árbol de Huffman.

class Huffman{

private:

map<char,int> frecuencies;

hNodo\* root;

public:

Huffman(map<char,int> frecuencies){

list<hNodo\*> stack,hojas;

for (auto i : frecuencies){

stack.push\_back( new hNodo(i.first, i.second) );

}

hojas = stack;

hNodo\* h1; hNodo\* h2; hNodo\* p;

while (stack.size() > 1){

stack.sort(comparePNodos);

h1 = stack.front(); stack.pop\_front();

h2 = stack.front(); stack.pop\_front();

p = \*h1 + \*h2;

h1->padre = p;

h2->padre = p;

stack.push\_front(p);

}

this->root = \*(stack.begin());

hNodo\* temp;

string codigo;

for (auto& i : hojas){

temp = i;

codigo = "";

while (temp != this->root){

if (temp->padre->izq == temp) codigo = codigo + "0";

if (temp->padre->der == temp) codigo = codigo + "1";

temp = temp->padre;

}

reverse(codigo.begin(),codigo.end());

i->camino = codigo;

}

for (auto i: hojas) this->keys[i->letra] = i ->camino;

}

void generateDot(string& res, hNodo\* t){

if (!t) return;

if (t->izq) res = res + t->id + " -> " + t->izq->id + "; \n";

if (t->der) res = res + t->id + " -> " + t->der->id + "; \n";

generateDot(res, t->izq);

generateDot(res, t->der);

}

void setIDs(hNodo\* t, int& i,string& res){

if (!t) return;

t->id = i;

res = res + i + " [ label = \"" + t->getStrValue() + "\" ]; \n";

i = i + 1;

setIDs(t->izq,i, res);

setIDs(t->der,i, res);

}

string getDot(){

string res = "digraph G {\n";

int i = 0;

this->setIDs(this->root, i, res);

this->generateDot(res, this->root);

res = res + "}";

return res;

}

string encriptar (string texto){

string res = "";

int i = 0;

while (texto[i] != '\0'){

res = res + this->keys[texto[i]];

i++;

}

return res;

}

};

* 1. Generar mapa de frecuencias: Eso se hace en el main, y se le manda a la clase Huffman para que genere el árbol.

int main(){

map<char,int> frecuencias;

string frase;

fstream file;

file.open("frase.txt");

string line;

while (!file.eof()){

getline(file,line);

frase = frase + line;

}

int i = 0;

while(frase[i] != '\0'){

if (frecuencias.find(frase[i]) != frecuencias.end()){

frecuencias[frase[i]]++;

}

else{

frecuencias[frase[i]]=1;

}

i++;

}

Huffman a(frecuencias);

cout<<"//"<<a.encriptar(frase)<<endl;

cout<<"//"<<a.desencriptar("100110")<<endl;

cout<<a.getDot()<<endl;

}

* 1. Desencriptar: La siguiente función desencripta usando el árbol de Huffman:

string desencriptar (string texto){

string res = "";

int i = 0;

hNodo\* temp;

while (texto[i] != '\0'){

temp = this->root;

while (!temp->hoja){

if (texto[i] == '0') temp = temp->izq;

if (texto[i] == '1') temp = temp->der;

i++;

}

if (temp) res = res + string(&temp->letra);

}

return res;

}

1. Resultado: El programa leerá el archivo frase.txt que tendrá el siguiente contenido.

*“a caballo regalado no se le mira el diente”*

Y el resultado encriptado de la misma frase será este:

11100011010111101010110100000111111111111001100000101101110000111110000011000111011100101110010111100001111011011000111110100111110001000111001001101101111010010001110111010011111000100011100111011101110001000101010110011001011110101100010

El resultado de encriptar le es: 100110.

Así mismo al desencriptar cualquiera de los dos resultados nos sale la palabra original.

Y este es el árbol de Huffman generado. En las hojas se encuentra el carácter junto con la frecuencia hallada y el código que se necesita para llegar ahí.

