Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

**Лабораторная работа №5**

**По дисциплине ЯП за 2 семестр**  
**Тема: «Структуры данных: Дерево»**

**Выполнил:**

Студент группы ПО-6(1)  
 1-го курса

Мартынович Даниил

**Проверила:**

Анфилец С.В

Брест, 2021

***Задание:***

1. Вспомнить: указатели, ссылки; структуры данных (записи); динамические структуры: стек,

дек; чтение данных из файла; функции: malloc, sizeof, free; new, delete.

2. Разобраться с алгоритмом кодирования Шеннона-Фано и Хеллмана.

3. Разобраться с принципами построения деревьев. Разработать подход построения

бинарного дерева, который реализует соответствующий (вашему варианту) алгоритм

кодирования.

4. Написать программу:

4.1. Считать текст из файла. Это исходный текст, на основании которого будет происходить

кодирование.

4.2. Составить статистику по символам, встречающимся в тексте, в отсортированном виде:

<Символ> <Частота>

… …

4.3. Реализовать необходимые типы/структуры для организации дерева. Разработать

функции для работы с деревьями. Протестировать их работу, прежде чем приступать к

реализации алгоритма кодирования.

4.4. Реализовать алгоритм построения дерева.

4.5. Отобразить (сохранить в файл) таблицу кодов для символов исходного текста.

***Алгоритмы оптимального кодирования.***

*1. Алгоритм Шеннона-Фано*

Один из первых алгоритмов сжатия, которые впервые сформулировали ученые Шеннон и

Фано, который использует коды переменной длины: часто встречающийся символ кодируется

кодом меньше длины, редко встречающийся — кодом большей длины. Коды Шеннона-Фано

префиксные, то есть никакое кодовое слово не является префиксом любого другого. Это

свойство позволяет однозначно декодировать любую последовательность кодовых слов.

Этапы:

1. Символы первичного алфавита выписывают по убыванию вероятностей (частоты).

2. Символы полученного алфавита делят на две части, суммарные вероятности которых

максимально близки друг к другу.

3. В префиксном коде для первой части алфавита присваивается двоичная цифра «0»,

второй части - «1». (первая часть — может интерпретироваться как левая ветка)

4. Полученные части рекурсивно делятся и их частям назначается соответствующие

двоичные цифры (до тех пор, пока количество элементов в части (в узле) не будет равно 1це).

Построение дерева для Алгоритма Шеннона-Фано начинается с корня, всё множество

кодируемых элементов соответствует корню дерева. Оно разбивается на два подмножества с

примерно одинаковыми суммарными вероятностями. Эти подмножества соответствуют двум

вершинам второго уровня, которые соединяются с корнем. Далее каждое их этих

подмножеств разбивается на два подмножества по тому же принципу. Им соответствуют

вершины третьего уровня. Если подмножество содержит единственный элемент, то ему

соответствует лист дерева. Таким образом выполняем алгоритм разбиения до тех пор пока не

получим все концевые вершины.

**Примечание.** Формируемый код можно хранить в каждом узле, таким образом, когда узел

будет содержать один символ (и останется листом), код хранящийся там будет

соответствовать реальному коду этого символа по алгоритму.

include <bits/stdc++.h>

using namespace std;

struct node {

string sym;

float pro;

int arr[20];

int top;

} p[20];

typedef struct node node;

// функция для поиска кода Шеннона

void shannon(int l, int h, node p[])

{

float pack1 = 0, pack2 = 0, diff1 = 0, diff2 = 0;

int i, d, k, j;

if ((l + 1) == h || l == h || l > h) {

if (l == h || l > h)

return;

p[h].arr[++(p[h].top)] = 0;

p[l].arr[++(p[l].top)] = 1;

return;

}

else {

for (i = l; i <= h - 1; i++)

pack1 = pack1 + p[i].pro;

pack2 = pack2 + p[h].pro;

diff1 = pack1 - pack2;

if (diff1 < 0)

diff1 = diff1 \* -1;

j = 2;

while (j != h - l + 1) {

k = h - j;

pack1 = pack2 = 0;

for (i = l; i <= k; i++)

pack1 = pack1 + p[i].pro;

for (i = h; i > k; i--)

pack2 = pack2 + p[i].pro;

diff2 = pack1 - pack2;

if (diff2 < 0)

diff2 = diff2 \* -1;

if (diff2 >= diff1)

break;

diff1 = diff2;

j++;

}

k++;

for (i = l; i <= k; i++)

p[i].arr[++(p[i].top)] = 1;

for (i = k + 1; i <= h; i++)

p[i].arr[++(p[i].top)] = 0;

// Вызов функции Шеннона

shannon(l, k, p);

shannon(k + 1, h, p);

}

}

// Функция для сортировки символов

// на основе их вероятности или частоты

void sortByProbability(int n, node p[])

{

int i, j;

node temp;

for (j = 1; j <= n - 1; j++) {

for (i = 0; i < n - 1; i++) {

if ((p[i].pro) > (p[i + 1].pro)) {

temp.pro = p[i].pro;

temp.sym = p[i].sym;

p[i].pro = p[i + 1].pro;

p[i].sym = p[i + 1].sym;

p[i + 1].pro = temp.pro;

p[i + 1].sym = temp.sym;

}

}

}

}

// функция для отображения кодов Шеннона

void display(int n, node p[])

{

int i, j;

cout << "\n\n\n\tSymbol\tProbability\tCode";

for (i = n - 1; i >= 0; i--) {

cout << "\n\t" << p[i].sym << "\t\t" << p[i].pro << "\t";

for (j = 0; j <= p[i].top; j++)

cout << p[i].arr[j];

}

}

int main()

{

int n, i, j;

float total = 0;

string ch;

node temp;

cout << "Enter number of symbols\t: ";

cin >> n;

for (i = 0; i < n; i++) {

cout << "Enter symbol " << i + 1 << " : ";

ch = (char)(65 + i);

cin >> ch;

p[i].sym += ch;

}

float x[] = { 0.22, 0.28, 0.15, 0.30, 0.05 };

for (i = 0; i < n; i++) {

cout << "\nEnter probability of " << p[i].sym << " : ";

cin >> x[i];

// Вставляем значение в узел

p[i].pro = x[i];

total = total + p[i].pro;

// проверка максимальной вероятности

if (total > 1) {

cout << "Invalid. Enter new values";

total = total - p[i].pro;

i--;

}

}

p[i].pro = 1 - total;

// Сортировка символов на основе

// их вероятность или частота

sortByProbability(n, p);

for (i = 0; i < n; i++)

p[i].top = -1;

// Найти код Шеннона

shannon(0, n - 1, p);

// Показать коды

display(n, p);

cout << endl;

system("pause");

return 0;

}

