МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Отчёт по лабораторной работе №6
по дисциплине
«Информатика»
«Кодирование информации.»

Разработал студент гр. ИВТб-1301-05-00 _____/Черкасов А. А./
(подпись)
Проверил доцент кафедры ЭВМ _____/Коржавина А.С./
(подпись)

Киров
2024

Цель работы

Цель работы: Изучить методы и алгоритмы кодирования информации. Реализовать равномерное кодирование и оптимальное кодирование с использованием алгоритмов Фано или Хаффмана, проанализировать их свойства и применимость в различных задачах.

Задания

1. **Равномерное кодирование.**

   Написать программу, выполняющую кодирование N различных состояний равномерно.

   Формат Ввода.

   Целое число N – количество различных состояний.

   Формат Вывода.

   Список кодов.

Ввод	Вывод
6	000 001 010 011 100 101

2. **Оптимальное кодирование.**

   Написать программу, выполняющую кодирование алгоритмом Хаффмана или Фано.

   Формат Ввода.

   Через пробел количество событий (кодовых состояний) и массив вероятностей каждого из событий.

   Формат Вывода.

   Коды состояний.

Ввод	Вывод
4 0.5 0.25 0.13 0.12	0 10 110 111

Решение

Схемы алгоритмов представлены на Рисунках 1.1, 1.2 и 1.3.

Код программ приведён в Приложениях А1 и А2.

Рисунок 1.1 - Схема алгоритма программы.

Вывод

В результате работы разработаны программы для равномерного и оптимального кодирования данных. Программы корректно обрабатывают входные данные и формируют коды в соответствии с заданными алгоритмами. Тестирование подтвердило их корректность и эффективность.

Приложение А1

```
#include <stdio.h>

void err() {printf("???\n");}

void gen(int n) {
    int bits = 0;
    while ((1 << bits) < n) {bits++;}
    for (int i = 0; i < n; i++) {
        for (int j = bits - 1; j >= 0; j--) {
            int bit = (i >> j) & 1;
            printf("%d", bit);
        }
        printf(" ");
    }
    printf("\n");
    return;
}

int main() {
    int n;
    scanf("%d", &n);
    if (n <= 0) {
        err();
        return 1;
    }
    gen(n);
    return 0;
}
```

Приложение А2

```
#include <stdlib.h>
#include <stdio.h>

typedef struct Node {
    float prob;
    struct Node *left, *right;
} Node;

Node* new(float prob) {
    Node* node = (Node*)malloc(sizeof(Node));
    node->prob = prob;
    node->left = NULL;
    node->right = NULL;
    return node;
}

void bubbleSort(Node* nodes[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (nodes[j]->prob > nodes[j + 1]->prob) {
                Node* temp = nodes[j];
                nodes[j] = nodes[j + 1];
                nodes[j + 1] = temp;
            }
        }
    }
}

Node* buildTree(Node* nodes[], int n) {
    while (n > 1) {
        bubbleSort(nodes, n);
```

```
        Node* left = nodes[1];
        Node* right = nodes[0];

        Node* newNode = (Node*)malloc(sizeof(Node));
        newNode->prob = left->prob + right->prob;
        newNode->left = left;
        newNode->right = right;

        nodes[0] = newNode;
        for (int i = 1; i < n - 1; i++) {
            nodes[i] = nodes[i + 1];
        }
        n--;
    }
    return nodes[0];
}

void printCodes(Node* root, int arr[], int top) {
    if (root->left) {
        arr[top] = 0;
        printCodes(root->left, arr, top + 1);
    }

    if (root->right) {
        arr[top] = 1;
        printCodes(root->right, arr, top + 1);
    }

    if (root->left == NULL && root->right == NULL) {
        for (int i = 0; i < top; i++) {
            printf("%d", arr[i]);
        }
        printf(": %.6f\n", root->prob);
```

```
    }
}

void freeTree(Node* root) {
    if (root) {
        freeTree(root->left);
        freeTree(root->right);
        free(root);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    printf("----------\n");
    Node* nodes[n];
    for (int i = 0; i < n; i++) {
        float prob;
        scanf("%f.6", &prob);
        nodes[i] = new(prob);
    }

    bubbleSort(nodes, n);
    Node* root = buildTree(nodes, n);
    int arr[100];
    int top = 0;
    printf("----------\n");
    printCodes(root, arr, top);
    freeTree(root);
    return 0;
}
```