

Модульна Контрольна Робота 2

Хіміч Богдан Вадимович ІПЗ 22 -3

Варіант 28

7. Алгоритм Шеннон-Фано (стиснення даних)

Алгоритм Шеннона-Фано - це алгоритм стиснення даних, розроблений Клодом Шенноном та Робертом Фано. Він використовується для зменшення розміру даних шляхом заміни часто зустрічаючих символів бітовими кодами, які складаються з меншої кількості бітів, тим самим забезпечуючи ефективніше представлення даних.

Опис алгоритму:

1. Починаємо з початкового набору символів, які ми хочемо стиснути.
2. Розраховуємо частоту появи кожного символу у наборі.
3. Сортуюмо символи за спаданням їхніх частот появи.
4. Рекурсивно ділимо набір символів на дві частини, розподіляючи символи таким чином, щоб сумарна частота появи в кожній частині була майже однаковою або дуже близькою.
5. Кожній половині присвоюється бітовий код: лівій половині - 0, правій половині - 1.
6. Рекурсивно застосовуємо кроки 4-5 до кожної половини, доки не досягнемо окремих символів.
7. Формуємо таблицю кодів, де для кожного символу ми зберігаємо його бітовий код.

Складність алгоритму Шеннона-Фано залежить від розподілу частот появи символів. В найкращому випадку, коли всі символи мають однакову частоту, складність становить $O(n \log n)$, де n - кількість символів у наборі. Однак, в загальному випадку, коли розподіл частот неоднорідний, складність може бути гіршою, до $O(n^2)$.

Структури даних, які використовуються в алгоритмі Шеннона-Фано:

1. Дерево Шеннона-Фано: Це бінарне дерево, де кожен внутрішній вузол представляє розділений набір символів. Лівий нащадок вузла представляє ліву частину розділеного набору символів, а правий нащадок - праву частину. Кожен листовий вузол містить окремий символ.
2. Таблиця кодів: Це структура даних, яка зберігає бітовий код, який відповідає кожному символу у наборі. Таблиця кодів може бути реалізована, наприклад, у вигляді асоціативного масиву, де символи виступають як ключі, а їхні бітові коди - як значення.
3. Масив частот: Це масив або список, який містить частоту появи кожного символу у вихідному наборі. Цей масив використовується для сортування символів та побудови дерева Шеннона-Фано.

Ці структури даних допомагають виконати сортування символів за частотою, рекурсивно розділити набір символів на піднабори, побудувати дерево Шеннона-Фано та зберегти таблицю кодів для подальшого використання при стисненні та розпакуванні даних за допомогою алгоритму Шеннона-Фано.

Важливо зазначити, що алгоритм Шеннона-Фано є одним з багатьох алгоритмів стиснення даних, і його ефективність залежить від конкретного розподілу частот символів у вихідному наборі даних.

Лістинг програми:

```
using System;
```

```
using System.Collections.Generic;
```

```
class ShannonFanoCompression
```

```
{
```

```
    private class Node
```

```
    {
```

```
        public char Symbol { get; set; }
```

```
        public string Code { get; set; }
```

```
    }
```

```
    private static List<Node> Encode(string data)
```

```
    {
```

```
        List<Node> nodes = new List<Node>();
```

```
        foreach (char c in data)
```

```
        {
```

```
            Node node = nodes.Find(n => n.Symbol == c);
```

```
            if (node != null)
```

```
            {
```

```
                node.Code += "1";
```

```
            }
```

```
            else
```

```
            {
```

```
                nodes.Add(new Node { Symbol = c, Code = "0" });
```

```
            }
```

```
    }
```

```

// Sort nodes by the length of their codes in descending order
nodes.Sort((n1, n2) => n2.Code.Length.CompareTo(n1.Code.Length));

Divide(nodes, 0, nodes.Count - 1);

return nodes;
}

private static void Divide(List<Node> nodes, int start, int end)
{
    if (start >= end)
        return;

    int sum = 0;
    for (int i = start; i <= end; i++)
    {
        sum += nodes[i].Code.Length;
    }

    int mid = FindPartition(nodes, start, end, sum / 2);

    for (int i = start; i <= mid; i++)
    {
        nodes[i].Code += "0";
    }

    for (int i = mid + 1; i <= end; i++)

```

```
{  
    nodes[i].Code += "1";  
}
```

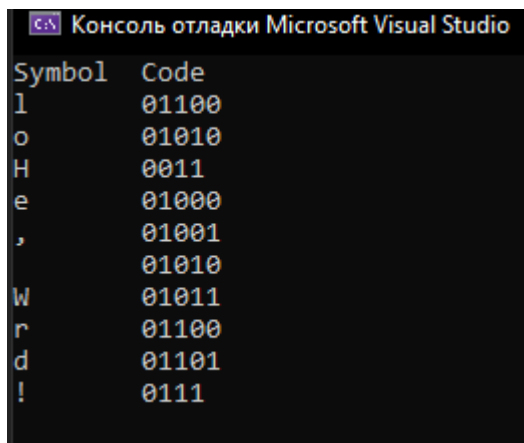
```
    Divide(nodes, start, mid);  
    Divide(nodes, mid + 1, end);  
}
```

```
private static int FindPartition(List<Node> nodes, int start, int end, int targetSum)  
{  
    int sum = 0;  
    for (int i = start; i < end; i++)  
    {  
        sum += nodes[i].Code.Length;  
        if (sum >= targetSum)  
        {  
            return i;  
        }  
    }  
    return start;  
}
```

```
static void Main()  
{  
    string data = "Hello, World!";  
    List<Node> encodedData = Encode(data);  
}
```

```
Console.WriteLine("Symbol\tCode");  
  
foreach (Node node in encodedData)  
{  
    Console.WriteLine($"{node.Symbol}\t{node.Code}");  
}  
}  
}
```

Скріншот лістингу програми:



Symbol	Code
l	01100
o	01010
H	0011
e	01000
,	01001
W	01010
r	01011
d	01100
!	01101
!	0111

Ця програма приймає вхідні дані (змінну data) і застосовує алгоритм Шеннона-Фано для кодування кожного символу у вхідних даних. Результатом є список символів та їх відповідних кодів. У цьому прикладі ми просто виводимо символи та їх коди на консоль.

Посилання на програму в gitlab: <https://gitlab.com/edroyze/rabota.git>