**Міністерство освіти і науки України**  
**Національний технічний університет України**  
**“Київський політехнічний інститут”**

Інститут Прикладного системного аналізу  
Кафедра Системного проектування

Лабораторна робота з дисципліни  
«Теорія інформації та кодування»

на тему  
**«СТИСНЕННЯ ДАНИХ В ДИСКРЕТНИХ ПОВІДОМЛЕННЯХ З ВИКОРИСТАННЯМ МЕТОДУ ШЕННОНА-ФАНО»**

Виконав:

студент 2 курсу

групи ДА-92

Насікан Д. Ю.

Київ 2021

**Мета роботи -** ознайомлення з методом ефективного кодування даних без втрат Шеннона-Фано, розробка алгоритму та програми для його реалізації, дослідження ефективності кодування даних у файлах різних типів і довільної довжини.

1. **ЗАВДАННЯ**

1. Ознайомитися з методикою ефективного кодування Шеннона-Фано.

2. Використовуючи табличний процесор MS Excel закодуйте методом Шеннона-Фано текстове повідомлення з вашим прізвищем ім'ям, по-батькові, датою і місцем народження. Підрахуйте ентропію текстового повідомлення, середню довжину кодових комбінацій закодованого повідомлення, коефіцієнт стиснення і оцініть ефективність кодування. Перевірте префіксність коду і розкодуйте повідомлення.

3. Розробити програму на мові програмування високого рівня для кодування і декодування текстових файлів довільної довжини. При вводі текстових даних передбачити два режиму вводу кодів: введення з клавіатури, а також введення даних з текстового файлу з використанням однобайтових кодових таблиць. За допомогою розробленої програми перевірити дані, отримані при виконанні п.2.

4. За допомогою розробленої при виконанні п.3 програми дослідити залежність коефіцієнту стиснення від довжини текстового файлу. Результати представити в табличній формі.

5. Модіфікувати програму, розроблену при виконанні п.4, для стиснення файлів довільних типів даних за алгоритмом Шеннона-Фано. Провести дослідження ефективності розробленої програми шляхом порівняння результатів стиснення текстових, мультимедійних і exe –файлів з допомогою розробленої програми і архіватора WinZip . Результати занести в таблицю.

1. **ХІД РОБОТИ**
   1. Використовуючи табличний процесор MS Excel закодуємо методом Шеннона-Фано текстове повідомлення з вашим прізвищем ім'ям, по-батькові, датою і місцем народження:

|  |  |  |
| --- | --- | --- |
| **Символ** | **Імовірність появи** | **Код** |
| **a** | **0.116** | **111** |
| **\_** | **0.093** | **1101** |
| **н** | **0.07** | **1100** |
| **2** | **0.07** | **1011** |
| **0** | **0.07** | **1010** |
| **с** | **0.047** | **1001** |
| **і** | **0.047** | **1000** |
| **и** | **0.047** | **0111** |
| **т** | **0.047** | **0110** |
| **р** | **0.047** | **0101** |
| **о** | **0.047** | **01001** |
| **в** | **0.047** | **01000** |
| **.** | **0.047** | **0011** |
| **1** | **0.047** | **00101** |
| **к** | **0.023** | **00100** |
| **д** | **0.023** | **00011** |
| **м** | **0.023** | **000101** |
| **ю** | **0.023** | **000100** |
| **й** | **0.023** | **00001** |
| **ч** | **0.023** | **000001** |
| **з** | **0.023** | **000000** |

Повідомлення: насікан дмитро юрійович 22.10.2001 заставна

Код:1100111100110000010011111001101000110001010111011001010100111010001000101100000001010010100001110000011101101110110011001011010001110111010101000101110100000011110010110111010001100111

Ентропія текстового повідомлення: 4.22

Середня довжина кодових комбінацій - 4.6.

З використанням однобайтових таблиць 1 символ повідомлення кодувався 8 бітами. Після кодування методом Шеннона-Фано 1 символ повідомлення кодується 4.6 бітами. Отже, коеф. стиснення – 1.74. Як бачимо, кодування досить ефективне, так як дозволяє використовувати коди, довжина яких майже в 2 рази менша.

Проаналізувавши таблицю кодів бачимо, що жодна кодова послідовність, яка кодує якийсь символ не є префіксом більш довгої комбінації. Отже, даний код є префіксним.

Після розкодування повідомлення отримуємо повідомлення: насікан дмитро юрійович 22.10.2001 заставна.

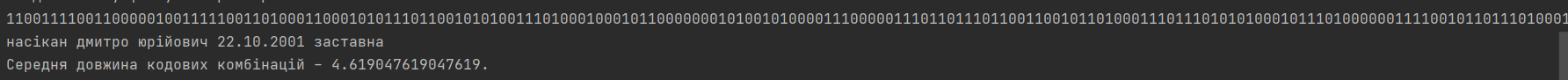
* 1. Розробимо програму на мові програмування високого рівня для кодування і декодування текстових файлів довільної довжини:

Лістинг програми:

# повертає відсортований список з частотою зустрічі кожного символа  
def construct\_frequency\_list(message):  
 result = {}  
 list\_result = []  
  
 for symbol in message:  
 try:  
 info = result[symbol]  
 except KeyError:  
 info = 0  
  
 info += 1  
 result[symbol] = info  
  
 for symbol in result.keys():  
 info = result[symbol]  
 info = [info, round(info/len(message), 3)]  
 list\_result.append([symbol, info])  
  
 return sorted(list\_result, key=lambda x: x[1][1], reverse=True)  
  
# обчислює суму починаючи з вибраного символа у вказану сторону  
def frequency\_sum(frequency\_list, start, upper\_direction):  
 result = 0  
  
 if upper\_direction is True:  
 while start >= 0:  
 result += frequency\_list[start][1][1]  
 start -= 1  
  
 else:  
 while start < len(frequency\_list):  
 result += frequency\_list[start][1][1]  
 start += 1  
  
 return round(result, 3)  
  
# знаходить серединний елемент (той, при якому суми рівні)  
def find\_mid(frequency\_list):  
 best\_result\_index = None  
 delta = 400000  
  
 for index, item in enumerate(frequency\_list):  
 upper\_sum = frequency\_sum(frequency\_list, index, True)  
 lower\_sum = frequency\_sum(frequency\_list, index + 1, False)  
  
 if delta > abs(upper\_sum - lower\_sum):  
 best\_result\_index = index  
 delta = abs(upper\_sum - lower\_sum)  
  
 return best\_result\_index  
  
def shannon\_fano\_code\_constructor(frequency\_list):  
 result\_dict = {}  
 \_\_shannon\_fano\_code\_constructor(frequency\_list, result\_dict)  
 return result\_dict  
  
def \_\_shannon\_fano\_code\_constructor(frequency\_list, codes\_dict):  
 mid = find\_mid(frequency\_list)  
  
 upper\_part = frequency\_list[:mid + 1]  
 lower\_part = frequency\_list[mid + 1:]  
  
 for item in upper\_part:  
 try:  
 code = codes\_dict[item[0]]  
 except KeyError:  
 code = ''  
  
 codes\_dict[item[0]] = code + '1'  
  
 for item in lower\_part:  
 try:  
 code = codes\_dict[item[0]]  
 except KeyError:  
 code = ''  
  
 codes\_dict[item[0]] = code + '0'  
  
 print(f'up: {upper\_part}\n low: {lower\_part}\n dict:{codes\_dict}\n\n\n')  
  
 if len(upper\_part) <= 1 and len(lower\_part) <= 1:  
 return  
  
 if len(lower\_part) <= 1:  
 \_\_shannon\_fano\_code\_constructor(upper\_part, codes\_dict)  
 return  
  
 if len(upper\_part) <= 1:  
 \_\_shannon\_fano\_code\_constructor(lower\_part, codes\_dict)  
 return  
  
 \_\_shannon\_fano\_code\_constructor(upper\_part, codes\_dict)  
 \_\_shannon\_fano\_code\_constructor(lower\_part, codes\_dict)  
  
def shannon\_fano\_encode(message):  
 f\_list = construct\_frequency\_list(message)  
 codes\_dict = shannon\_fano\_code\_constructor(f\_list)  
  
 encoded\_message = ''  
 for symbol in message:  
 encoded\_message += codes\_dict[symbol]  
  
 return encoded\_message, codes\_dict  
  
def make\_code\_to\_sym\_dict(sym\_to\_code\_dict):  
 result = {}  
 for key, val in sym\_to\_code\_dict.items():  
 result[val] = key  
  
 return result  
  
def shannon\_fano\_decode(enc\_message, codes\_dict):  
 buffer = ''  
 decoded\_message = ''  
 for symbol in enc\_message:  
 buffer += symbol  
  
 try:  
 decoded\_symbol = codes\_dict[buffer]  
 except KeyError:  
 continue  
 decoded\_message += decoded\_symbol  
 buffer = ''  
  
 return decoded\_message  
  
def average\_code\_length(sym\_to\_code\_dict):  
 result = 0  
 for code in sym\_to\_code\_dict.values():  
 result += len(code)  
  
 return result / len(sym\_to\_code\_dict.values())  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 MODE = int(input('Виберіть режим роботи:\n1 - текстовий файл\n2 - введення повідомлення вручну\n\n'))  
 message = None  
 sym\_to\_code\_dict = None  
  
 if MODE == 1:  
 file\_name = input('Уведіть назву файлу без розширення: ') + '.txt'  
 with open(file\_name, 'r', encoding='utf-8') as file:  
 text\_message = file.read()  
 message, sym\_to\_code\_dict = shannon\_fano\_encode(text\_message)  
  
 elif MODE == 2:  
 text\_message = input('Введіть ваше текстове повідомлення: ')  
 message, sym\_to\_code\_dict = shannon\_fano\_encode(text\_message)  
  
 code\_to\_sym\_dict = make\_code\_to\_sym\_dict(sym\_to\_code\_dict)  
 print(message)  
 print(sym\_to\_code\_dict)  
 print(code\_to\_sym\_dict)  
  
 print(shannon\_fano\_decode(message, code\_to\_sym\_dict))  
 print(f'Середня довжина кодових комбінацій - {average\_code\_length(sym\_to\_code\_dict)}.')

Перевіримо результати пункту 2.1 на правильність, користуючись програмою:





Отриманий код: 1100111100110000010011111001101000110001010111011001010100111010001000101100000001010010100001110000011101101110110011001011010001110111010101000101110100000011110010110111010001100111.

Розкодоване повідомлення: насікан дмитро юрійович 22.10.2001 заставна

Середня довжина кодових комбінацій: 4.619047619047619.

* 1. За допомогою розробленої при виконанні п.2.2 програми дослідимо залежність коефіцієнту стиснення від довжини текстового файлу:

|  |  |  |
| --- | --- | --- |
| **Довжина текстового файлу** | **Коефіцієнт стиснення** | **Середня довжина кодових комбінацій** |
| **43** | **1.73** | **4.61** |
| **196** | **1.68** | **6.075** |
| **619** | **1.64** | **6.85** |
| **1024** | **1.63** | **7.41** |
| **1694** | **1.58** | **7.69** |
| **4493** | **1.58** | **8.8** |

Як бачимо, що більша довжина файлу, то меншим стає коефіцієнт стиснення.

* 1. Модіфікуємо програму, розроблену при виконанні п.4, для стиснення файлів довільних типів даних за алгоритмом Шеннона-Фано:

# повертає відсортований список з частотою зустрічі кожного символа  
def construct\_frequency\_list(message):  
 result = {}  
 list\_result = []  
  
 for symbol in message:  
 try:  
 info = result[symbol]  
 except KeyError:  
 info = 0  
 info += 1  
 result[symbol] = info  
  
 for symbol in result.keys():  
 info = result[symbol]  
 info = [info, round(info/len(message), 3)]  
 list\_result.append([symbol, info])  
  
 return sorted(list\_result, key=lambda x: x[1][1], reverse=True)  
  
# обчислює суму починаючи з вибраного символа у вказану сторону  
def frequency\_sum(frequency\_list, start, upper\_direction):  
 result = 0  
  
 if upper\_direction is True:  
 while start >= 0:  
 result += frequency\_list[start][1][1]  
 start -= 1  
  
 else:  
 while start < len(frequency\_list):  
 result += frequency\_list[start][1][1]  
 start += 1  
  
 return round(result, 3)  
  
# знаходить серединний елемент (той, при якому суми рівні)  
def find\_mid(frequency\_list):  
 best\_result\_index = None  
 delta = 400000  
  
 for index, item in enumerate(frequency\_list):  
 upper\_sum = frequency\_sum(frequency\_list, index, True)  
 lower\_sum = frequency\_sum(frequency\_list, index + 1, False)  
  
 if delta > abs(upper\_sum - lower\_sum):  
 best\_result\_index = index  
 delta = abs(upper\_sum - lower\_sum)  
  
 return best\_result\_index  
  
def shannon\_fano\_code\_constructor(frequency\_list):  
 result\_dict = {}  
 \_\_shannon\_fano\_code\_constructor(frequency\_list, result\_dict)  
 return result\_dict  
  
def \_\_shannon\_fano\_code\_constructor(frequency\_list, codes\_dict):  
 mid = find\_mid(frequency\_list)  
  
 upper\_part = frequency\_list[:mid + 1]  
 lower\_part = frequency\_list[mid + 1:]  
  
 for item in upper\_part:  
 try:  
 code = codes\_dict[item[0]]  
 except KeyError:  
 code = ''  
  
 codes\_dict[item[0]] = code + '1'  
  
 for item in lower\_part:  
 try:  
 code = codes\_dict[item[0]]  
 except KeyError:  
 code = ''  
  
 codes\_dict[item[0]] = code + '0'  
  
  
 if len(upper\_part) <= 1 and len(lower\_part) <= 1:  
 return  
  
 if len(lower\_part) <= 1:  
 \_\_shannon\_fano\_code\_constructor(upper\_part, codes\_dict)  
 return  
  
 if len(upper\_part) <= 1:  
 \_\_shannon\_fano\_code\_constructor(lower\_part, codes\_dict)  
 return  
  
 \_\_shannon\_fano\_code\_constructor(upper\_part, codes\_dict)  
 \_\_shannon\_fano\_code\_constructor(lower\_part, codes\_dict)  
  
def shannon\_fano\_encode(message):  
 f\_list = construct\_frequency\_list(message)  
 codes\_dict = shannon\_fano\_code\_constructor(f\_list)  
  
 encoded\_message = ''  
 for symbol in message:  
 encoded\_message += codes\_dict[symbol]  
  
 return encoded\_message, codes\_dict  
  
def make\_code\_to\_sym\_dict(sym\_to\_code\_dict):  
 result = {}  
 for key, val in sym\_to\_code\_dict.items():  
 result[val] = key  
  
 return result  
  
def shannon\_fano\_decode(enc\_message, codes\_dict):  
 print(codes\_dict)  
 buffer = ''  
 decoded\_message = b''  
 for symbol in enc\_message:  
 print(buffer)  
 print(symbol)  
 buffer += symbol  
  
 try:  
 decoded\_symbol = codes\_dict[buffer]  
 except KeyError:  
 continue  
 print(decoded\_symbol)  
 hex\_str = hex(decoded\_symbol)  
 print(hex\_str)  
 decoded\_message += bytes.fromhex(hex\_str[2:])  
 buffer = ''  
  
 return decoded\_message  
  
def average\_code\_length(sym\_to\_code\_dict):  
 result = 0  
 for code in sym\_to\_code\_dict.values():  
 result += len(code)  
  
 return result / len(sym\_to\_code\_dict.values())  
  
def write\_file(encoded\_message):  
 bytes\_array = []  
  
 for i in range(len(encoded\_message)//8):  
 bytes\_array.append(int(encoded\_message[i \* 8: i \* 8 + 8], 2))  
  
  
 with open(f'out.lab1', 'wb') as f:  
 for i in bytes\_array:  
 f.write(bytearray(i))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 file\_name = input('Уведіть назву файлу з розширенням: ')  
 with open(file\_name, 'rb') as file:  
 text\_message = file.read()  
 message, sym\_to\_code\_dict = shannon\_fano\_encode(text\_message)  
  
 code\_to\_sym\_dict = make\_code\_to\_sym\_dict(sym\_to\_code\_dict)  
  
 code\_len = average\_code\_length(sym\_to\_code\_dict)  
 print(f'Середня довжина кодових комбінацій - {code\_len}.')  
 print(f'Коефіцієнт стиснення - {8/code\_len}.')  
 print(f'Довжина файлу - {len(text\_message)}')  
  
 write\_file(message)

Протестуємо програму та зведемо дані в таблицю порівняння:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вхідний файл** | | | **Результат роботи розробленої програми** | | | **Архіватор WinRAR** | | |
| **Назва** | **Розмір файлу** | **Ентропія** | **Розмір файлу** | **Ентропія** | **Коефіцієнт стиснення** | **Розмір файлу** | **Ентропія** | **Коефіцієнт стиснення** |
| **5.jpg** | **147 KB** | **7.98** | **148 KB** | **7.99** | **0.99** | **147 KB** | **7.99** | **1** |
| **6.mp3** | **258 KB** | **7.69** | **251 KB** | **7.99** | **1.027** | **224 KB** | **7.99** | **1.1517** |
| **7.exe** | **217 KB** | **6.37** | **177 KB** | **7.88** | **1.226** | **101 KB** | **7.99** | **2.03** |
| **8.pdf** | **619 KB** | **7.99** | **619 KB** | **7.99** | **1** | **619 KB** | **7.99** | **1** |

1. **ВИСНОВКИ**

У ході даної лабораторної роботи було розглянуто метод ефективного кодування Шеннона-Фано, та розроблена програма, що реалізує цей алгоритм кодування для текстових файлів. Як приклад, було взято повідомлення «насікан дмитро юрійович 22.10.2001 заставна», у результаті перекодування, отримали коефіцієнт стиснення 1.73.

Для тестування програми було взято файли різної довжини, як можна помітити з таблиці, коефцієнт стиснення майже не залежить від довжини файлу.

У другій частині роботи програма була модифікованою під усі типи файлів. Тепер файл зчитувався у бінарному режимі, і методом Шеннона-Фено кодувалися байти повідомлення. Як бачимо з порівняльної таблиці, алгоритм найкраще стискає файли формати exe.