Лабораторна робота 2
Кодування даних алгоритмом Лемпеля-Зіва-Велча

Мета роботи: вивчення кодування та декодування інформації алгоритмом Лемпеля-Зіва-Велча (LZW) для усунення надлишковості даних при шифруванні.

1. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Надлишковість даних.

Характерною особливістю більшості типів даних є їх надлишковість. Для людини надлишковість даних часто пов'язана з якістю інформації, оскільки надлишковість, як правило, покращує зрозумілість та сприйняття інформації. Однією з форм кодування при якій з тексту видаляється надлишковість (надмірність) є *стиснення даних*. Стиснення (архівація) сприяє шифруванню й утруднює пошук шифру статистичним методом. Основний принцип, на якому базується стиснення даних, полягає в економічному описі повідомлення, згідно якому можливе відновлення початкового його значення з похибкою, яка контролюється.

В сучасному криптоаналізі, тобто науці про протистояння криптографії, з очевидністю доведено, що ймовірність злому криптосхеми при наявності кореляції між блоками вхідної інформації значно вище, ніж при відсутності такої. А алгоритми стиснення даних за визначенням і мають своїм основним завданням усунення надмірності, тобто кореляцій між даними у вхідному тексті.

Різні методи оптимального кодування базуються на зменшенні надлишковості викликаної неоднаковою апріорною ймовірністю символів або залежністю між порядком надходження символів.

На даний час існує величезна кількість програм для стиснення даних, заснованих на декількох основних способах.

Всі алгоритми стиснення даних якісно діляться на:

1) алгоритми стиснення без втрат, при використанні яких дані на приймаючій стороні відновлюються без найменших змін;

2) алгоритми стиснення з втратами, які видаляють з потоку даних інформацію, що не надто сильно впливає на суть даних, або взагалі не сприймається людиною (такі алгоритми зараз розроблені тільки для аудіо та відео зображень).

В криптосистемах, природно, використовується тільки перша група алгоритмів.

Існує декілька основних методів стиснення без втрат, зокрема:

• алгоритм Лемпеля-Зива (англ. Lempel, Ziv), орієнтований на стиск будь-яких видів текстів, тобто використовує факт неодноразового повторення "слів" - послідовностей байт. Алгоритм Лемпеля-Зива заснований на кореляціях між розташованими поруч символами алфавіту (словами, керуючими послідовностями, заголовками файлів фіксованою структури);

• алгоритм Хаффмана (англ. Huffman), орієнтований на стиск послідовностей байт, не пов'язаних між собою.

Практично всі популярні програми архівації без втрат (ARJ, RAR, ZIP тощо) використовують об'єднання цих двох методів - алгоритм LZH.

Існує багато практичних алгоритмів стиснення даних. Однак, в основі цих методів лежать чотири теоретичних алгоритми: алгоритм RLE (Run Length Encoding); алгоритми групи KWE (KeyWord Encoding); алгоритм Хафмана; сімейство алгоритмів LZ78 (LZW, MW, AP, Y)

В основі алгоритму RLE лежить ідея виявлення послідовностей даних, що повторюються, та заміни цих послідовностей більш простою структурою, в якій вказується код даних та коефіцієнт повторення. В основі алгоритму KWE покладено принцип кодування лексичних одиниць групами байт фіксованої довжини. Результат кодування зводиться в таблицю, утворюючи так званий словник. В основі алгоритму Хафмана лежить ідея кодування бітовими групами. Після

частотного аналізу вхідної послідовності символи сортуються за спаданням частоти входження. Чим частіше зустрічається символ, тим меншою кількістю біт він кодується. Результат кодування зводиться в словник, що необхідний для декодування.

Методи стиснення даних групуються в залежності від характеру інформації, що стискається: стиснення символьної інформації (документів), стиснення нерухомих графічних зображень (JPEG) і стиснення рухомих графічних зображень (MPEG). Основним параметром, що характеризує рівень стиснення інформації є коефіцієнт стиснення, який визначається відношенням обсягу не стиснених даних до обсягу стиснених (наприклад 5:1).

1.2. Метод LZW.

Відомі методи оптимального кодування різнозначними кодами (кодами різної довжини), які базуються на врахуванні різної частоти символів в тексті. Найбільш відомі з них: метод Шеннона-Фано та метод Хаффмана. Ці методи забезпечують достатньо ефективне стиснення без втрат, проте для підвищення рівня стиснення даних доцільно перейти від оптимального кодування окремих символів до кодування буквосполучень. Саме таке кодування забезпечує метод LZW (Лемпеля-Зіва-Велча). Алгоритм відрізняється високою швидкістю роботи при кодуванні та декодуванні, невибагливістю до пам’яті та простотою апаратної реалізації. Недолік – менший коефіцієнт компресії порівняно зі схемою двоступеневого кодування.

Цей алгоритм використовує ідею адаптивного кодування – за один прохід тексту одночасно будується динамічно словник та кодується текст. При цьому словник не зберігається. За рахунок того, що при декодуванні використовується той самий алгоритм побудови словника, словник автоматично відновлюється. Словник будується під час кодування наступним чином: певним послідовностям символів (словам) ставляться у відповідність групи бітів фіксованої довжини (звичайно 12-бітні). Словник ініціалізується усіма 1-символьними рядками (у випадку 8-бітових рядків – це 256 записів). Під час кодування, алгоритм переглядає текст символ за символом, та зберігає кожний новий, унікальний 2-символьний рядок у словник у вигляді пари код/символ, де код посилається у словнику на відповідний перший символ. Після того, як новий 2-символьний рядок збережений у словнику, на вихід передається код першого символу. Коли на вході зчитується черговий символ, для нього у словнику знаходиться рядок, який вже зустрічався і який має максимальну довжину, після чого у словнику зберігається код цього рядка з наступним символом на вході; на вихід видається код цього рядка, а наступний символ використовується в якості початку наступного рядка.

- Приклад кодування фрази **ABBACBBCAC**

Таблиця 1

Рядок	Символ	Вихід	Словник (Код)
			1 = A
			2 = B
			3 = C
A	B	1	4 = AB
B	B	2	5 = BB
B	A	2	6 = BA
A	C	1	7 = AC
C	B	3	8 = CB
B	B		
BB	C	5	9 = BBC
C	A	3	10 = CA
A	C		
AC		7	

В результаті отримуємо послідовність: **1 2 2 1 3 5 3 7**

- Приклад кодування послідовності: **1 2 2 1 3 5 3 7**

Таблиця 2

Попередній символ послідовності	Новий символ послідовності	Символ	Вихід	Словник (Код)
				1 = A
				2 = B
				3 = C
	1	A	A	
1	2	B	B	4 = AB
2	2	B	B	5 = BB
2	1	A	A	6 = BA
1	3	C	C	7 = AC
3	5	B	BB	8 = CB
5	3	C	C	9 = BBC
3	7	A	AC	10 = CA
7				

В результаті отримуємо послідовність: **A B B A C B B C A C**

2. ЛАБОРАТОРНЕ ЗАВДАННЯ

РОЗРОБИТИ ПРОГРАМУ ДЛЯ ВИКОНАННЯ КОДУВАННЯ ТА ДЕКОДУВАННЯ ЗАДАНОЇ ПОСЛІДОВНОСТІ, З ВИВЕДЕННЯМ НА ЕКРАН ОСНОВНИХ ЧАСТИН (ВХОДІВ ТА ВИХОДІВ, СЛОВНИКА)

1. Ознайомтесь з методом LZW.
2. Для заданого варіанту виконайте кодування заданого тексту
3. Виконайте декодування результуючої послідовності

3. ОФОРМЛЕННЯ ЗВІТУ

1. Результати кодування та декодування тексту методами LZW для заданого варіанту.

ВАРІАНТИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ

Стиснути заданий текст методом LZW. Показати як відновлюється початковий текст.

Варіант 1	ЛІМПОПОМПОПОНІ
Варіант 2	АНАШАНАШАНАШАФІ
Варіант 3	ГАННАНАНАГАНІАНА
Варіант 4	ТАНАНАКАНАНАКТАН
Варіант 5	ПОПОНАНІОПОНІОПА
Варіант 6	МОМОНОМАМОМОНО
Варіант 7	НООНОТОПОТОТОПООП
Варіант 8	ЛАЛАМЕЛАМЕЛАЛАМЕ
Варіант 9	ВАЛАВАЛЕВАЛЕЛЕВА
Варіант 10	КАЛЕВАЛАВАЛКАКЕКЕВ
Варіант 11	РАДОГОМУДОРОДОГАДО
Варіант 12	НАПАВАПАНАВАНАПА
Варіант 13	ЛОЛОВАВАЛОВАВАЛОЛО
Варіант 14	ТАТАМІМІТАТАМІТАТА
Варіант 15	ЖОЖОБАБАЖЖОЖОЖОБА
Варіант 16	АМАМАЛАМАЛОМАМАЛИ
Варіант 17	ВАКАРОРОКАВАКАРОВА
Варіант 18	НАНАГАГАНАГАНАНА
Варіант 19	ШАНАНАШАШАААШАНА
Варіант 20	ШАКАЛКАЛАШЛАШКА
Варіант 21	КАЛАННАЛАКЛАККАЛА
Варіант 22	ВАДАДАВАЛАВАЛАДА
Варіант 23	ЕГЕРРЕГЕНТЕНТРЕГЕГЕ
Варіант 24	УЛУКАНКАНАЛУКАНАЛУК
Варіант 25	ПРОРОПОРОПРОПОРОПО