национальный исследовательский университет "Высшая школа экономики"

Контрольное домашнее задание по дисциплине "Алгоритмы и структуры данных"

Выполнила: Манахова Мария,
студентка группы БПИ184(1)

Москва, 2020

Содержание

1 Постановка задачи

1. Разработать на языке C++ программу, реализующую алгоритмы сжатия данных без потерь, получение архивного файла из исходного и разархивированного файла их архивного (упаковка файла и распаковка архива)

   1) алгоритм Хаффмана (простой)
   2) алгоритм Лемпеля-Зива LZ77 (со скользящим окном)

2. Провести вычислительный эксперимент для исследования эффективности реализованных алгоритмов сжатия без потерь для файлов разного типа. Для проведения эксперимента с алгоритмами сжатия без потерь необходимо использовать набор из **N** файлов различных типов с именами 1.* – N.*.

   В наборе должны присутствовать файлы:

   - текстовый .txt
   - документ Word .docx
   - презентация .pptx
   - документ .pdf
   - исполняемый файл .exe или библиотечный .dll
   - цветное изображение .jpg
   - изображение черно-белое или в градациях серого .jpg
   - цветное изображение .bmp
   - изображение черно-белое или в градациях серого .bmp
   - файлы других форматов.

   Все исходные файлы должны быть помещены в папку DATA, находящейся там же, где и исполняемый файл. Программа должна автоматически обрабатывать все файлы из этой папки и в нее же записывать полученные результаты.

   Форматы имен файлов:

   1) исходный файл <name>.*
   2) метод упаковки, использующий алгоритм Хаффмана, упакованный файл <name>.haff
   3) метод распаковки, использующий алгоритм Хаффмана, распакованный файл <name>.unhaff
   4) метод упаковки, использующий алгоритм LZ77
      - размер скользящего окна 5 Кб, размер словаря 4 Кб, архивированный файл <name>.lz7705
      - размер скользящего окна 10 Кб, размер словаря 8 Кб, архивированный файл <name>.lz7710
      - размер скользящего окна 20 Кб, размер словаря 10 Кб, архивированный файл <name>.lz7720

5) метод распаковки, использующий алгоритм LZ77
   - размер скользящего окна 5 Кб, размер словаря 4 Кб, разархивированный файл <name>.unlz7705
   - размер скользящего окна 10 Кб, размер словаря 8 Кб, разархивированный файл <name>.unlz7710
   - размер скользящего окна 20 Кб, размер словаря 10 Кб, разархивированный файл <name>.unlz7720

Вычислить:

1) *энтропию* исходных файлов. Определяется общее количество различных символов w, вычисляется их частотная встречаемость w_i в файле, и энтропия файла по формуле:

$$H = -\sum_{i=1}^{m} w_i \cdot log_2 w_i$$

2) коэффициент сжатия как отношение размера сжатого файла к размеру исходного файла.

Измерить для каждого файла и алгоритма:

1) время упаковки;
2) время распаковки.

Время измерять в тактах ЦП или в наносекундах. Для получения достоверных результатов упаковку и распаковку каждого файла каждым методом выполнить не менее 10 раз, после чего вычислить среднее время работы каждого алгоритма на каждом файле.

3. Подготовить отчет по итогам работы, содержащий постановку задачи с указанием выполненных и невыполненных пунктов, описание алгоритмов и использованных структур данных, описание реализации алгоритмов, план эксперимента, описание аппаратных средств, результаты эксперимента, сравнительный анализ алгоритмов по эффективности сжатия файлов разных типов и по скорости работы, основные выводы и список использованных источников.

Выполненные пункты работы:

1. Реализованы архивация и разархивация файлов алгоритмами Хаффмана и LZ77
2. Построена таблица с коэффициентами сжатия файлов
3. Построена таблица со временем упаковки и распаковки файлов
4. Построены столбчатые диаграммы, отражающие:
   - коэффициент сжатия каждого файла для каждого алгоритма
   - время упаковки каждого файла для каждого алгоритма
   - время распаковки каждого файла для каждого алгоритма
5. Сделаны выводы по работе.

2 Описание алгоритмов, их реализации и использованных структур данных

2.1 Алгоритм Хаффмана

Алгоритм Хаффмана – алгоритм оптимального префисного кодирования алфавита с минимальной избыточностью. Алгоритм работает в два прохода: сначала строится таблица частот встречаемости символов во входном файле. Затем строится дерево частот и таблица кодов. Таблица частот записывается в выходной файл. Затем по таблице частот строится таблица соответствия символов и их битовых кодов, после чего осуществляется проход по входному файлу и каждый прочитанный символ кодируется его битовым кодом, а затем записывается в упакованный файл.

Код Хаффана строится с помощью дерева. По каждому символу строятся листья дерева, причем каждый из листьев имеет вес, равный количеству вхождений этого символа в сжимаемый файл. После этого выбираются две вершины с наименьшими весами, объединяются в родительскую вершину с частотой, равной сумме частот этих вершин. Родитель добавляется в список свободных вершин, а его потомки удаляются оттуда. Эти шаги повторяются, пока в списке не останется только одна свободная вершина – корень дерева. Далее по дереву рекурсивно строится таблица кодов символов.

При распаковке алгоритм считывает таблицу частот, записанную в начале файла и строит таблицу перевода символов, после чего осуществляется проход по архивированному файлу. Когда прочитанные биты образуют код, содержащийся в таблице перевода символов, символ декодируется и записывается в распакованный файл.

Структуры данных:

- std::vector<T> для хранения дерева/таблицы частот и буфера файла
- std::map<T> для хранения таблицы кодов

2.2 Алгоритм LZ77

Алгоритм LZ77 – алгоритм сжатия без потерь, основанный на поиске повторов фрагментов текста в данных и замене повторов ссылкой (кодом) на первое (или предыдущее) вхождение этого фрагмента в данные. Основан на принципе динамического словаря. Алгоритм работает за один проход: одновременно динамически строится словарь и кодируется файл. При этом словарь не хранится – за счет того, что при декодировании используется тот же самый алгоритм построения словаря, словарь динамически восстанавливается.

Алгоритм ищет в буфере предыстории фрагмент текста (файла) максимальной длины из буфера предпросмотра. На выход выдается код – тройка значений: позиция фрагмента в буфере предыстории, длина фрагмента, первый символ буфера предпросмотра после найденного фрагмента. Если фрагмент не найден, выдается «нулевая фраза»: берется код с нулевой позицией фрагмента в буфере предыстории и длиной фрагмента и следующий символ буфера. Таким образом кодируются все данные файла.

При распаковке алгоритм считывает коды-тройки, декодирует их и записывает полученную информацию в распакованный файл.

Структуры данных:

- std::vector<T> для хранения кодов-троек
- std::string для удобного хранения буфера файла, так как необходимо брать подстроки

3 План эксперимента

В программе задаются директория с тестируемыми файлами, имена этих файлов, а также директория для таблицы с результатами. Программа проходит по всем тестируемым файлам в директории и выполняет их упаковку и распаковку с помощью всех четырех алгоритмов: Хаффмана, LZ77 с размером скользящего окна 5Кб и размером словаря 4Кб, LZ77 с размером скользящего окна 10Кб и размером словаря 8Кб, LZ77 с размером скользящего окна 20Кб и размером словаря 16Кб. Помимо этого, программа также считает энтропию каждого из входных файлов, степень сжатия файлов для каждого из алгоритмов, время упаковки и распаковки файлов для каждого из алгоритмов, и записывает эти данные в файл results.csv. Для каждого файла создается 8 файлов – 4 упакованных файла с расширениями ".haff", ".lz7705", ".lz7710", ".lz7720" и 4 упакованных файла с расширениями ".unhaff", ".unlz7705", ".unlz7710", ".unlz7720". Программа запускается 10 раз, после каждого запуска данные копируются в отдельный файл в программе Excel, затем вручную усредняются и формируется итоговая таблица с результатми эксперимента, также с помощью Excel рисуются графики зависимостей.

4 Описание аппаратных средств и инструментов для разработки

Разработка и тестирование производились на ноутбуке 13-inch MacBook Pro 2019 с 4-ядерным процессором Intel Core i5 8-ого поколения с тактовой частотой 1.4 ГГц, 8 ГБ оперативной памяти и операционной системой MacOS. Для написания и отладки программы была использована среда разработки CLion 2019.3. Для сборки был использован CMake. Для компиляции использовался компилятор gcc 4.2.1. Программа написана с использованием языка C++ 17.

5 Результаты эксперимента

5.1 Таблицы

имя файла	энтропия	размер исходного файла, Мб
1.txt	4.498920	5
2.docx	7.994702	4.9
3.pptx	7.872054	4.9
4.pdf	7.438720	4.8
5.exe	7.919497	4.9
6.jpg	7.980790	5.2
7.jpg	7.979070	4.7
8.bmp	6.968415	4.3
9.bmp	7.345168	4.2
10.avi	7.574956	5.1

Таблица 1. Энтропия и размер исходных файлов

файл	Алгоритм Хаффмана			
	размер сжатого файла, Мб	коэффициент сжатия	время упаковки, с	время распаковки, с
1.txt	2.8	0.568037	0.431347	4.156234
2.docx	4.9	1.000263	0.700833	33.079324
3.pptx	4.9	0.987583	0.726940	33.638017
4.pdf	4.5	0.934999	0.660786	28.564938
5.exe	4.8	0.993327	0.722820	31.966569
6.jpg	5.2	0.999810	0.775822	35.617996
7.jpg	4.7	1.000059	0.706308	31.785455
8.bmp	3.7	0.875379	0.511925	21.205588
9.bmp	3.9	0.922590	0.534283	23.024098
10.avi	5.1	0.950964	1.506297	34.946787

Таблица 2.1. Коэффициент сжатия, размер сжатого файла, время упаковки и распаковки с использованием алгоритма Хаффмана

файл	Алгоритм LZ77, окно 5Кб			
	размер сжатого файла, Мб	коэффициент сжатия	время упаковки, с	время распаковки, с
1.txt	8.4	1.684784	69.482248	0.115214
2.docx	21	4.307088	163.029842	0.247243
3.pptx	17.8	3.594148	169.179161	0.211668
4.pdf	15.1	3.163832	111.606330	0.172715
5.exe	19.1	3.933537	128.917025	0.233972
6.jpg	22.7	4.323367	199.526828	0.261013
7.jpg	20.4	4.311656	152.954087	0.238690
8.bmp	12.3	2.873658	77.165197	0.230304
9.bmp	4	0.946209	24.777269	0.057967
10.avi	17.7	3.500678	140.970467	0.230124

Таблица 2.2. Коэффициент сжатия, размер сжатого файла, время упаковки и распаковки с использованием алгоритма LZ77 с размером буфера предпросмотра 5Кб

файл	Алгоритм LZ77, окно 10Кб			
	размер сжатого файла, Мб	коэффициент сжатия	время упаковки, с	время распаковки, с
1.txt	7.5	1.512138	71.125257	0.104739
2.docx	20.4	4.182156	168.176939	0.240483
3.pptx	16.6	3.354893	152.625582	0.200409
4.pdf	14.6	3.056296	112.613990	0.179213
5.exe	18.5	3.813477	124.006412	0.220423
6.jpg	22	4.185489	199.126667	0.259922
7.jpg	19.7	4.170699	150.371577	0.237177
8.bmp	10.1	2.362863	67.217598	0.131173
9.bmp	3.8	0.899746	25.092380	0.056770
10.avi	16.8	3.322750	136.889679	0.208807

Таблица 2.2. Коэффициент сжатия, размер сжатого файла, время упаковки и распаковки с использованием алгоритма LZ77 с размером буфера предпросмотра 10Кб

файл	Алгоритм LZ77, окно 20Кб			
	размер сжатого файла, Мб	коэффициент сжатия	время упаковки, с	время распаковки, с
1.txt	7.3	1.458305	68.476404	0.098508
2.docx	20.2	4.127874	157.508072	0.238536
3.pptx	16.4	3.307760	143.358404	0.200697
4.pdf	14.4	3.017213	111.254543	0.175841
5.exe	18.3	3.763619	129.668962	0.223847
6.jpg	21.6	4.124500	191.022913	0.263584
7.jpg	19.4	4.101490	151.453339	0.244783
8.bmp	9.4	2.211305	63.566950	0.126726
9.bmp	3.7	0.883556	24.189958	0.064881
10.avi	16.5	3.263120	132.741841	0.205175

Таблица 2.2. Коэффициент сжатия, размер сжатого файла, время упаковки и распаковки с использованием алгоритма LZ77 с размером буфера предпросмотра 20Кб

5.2 Графики

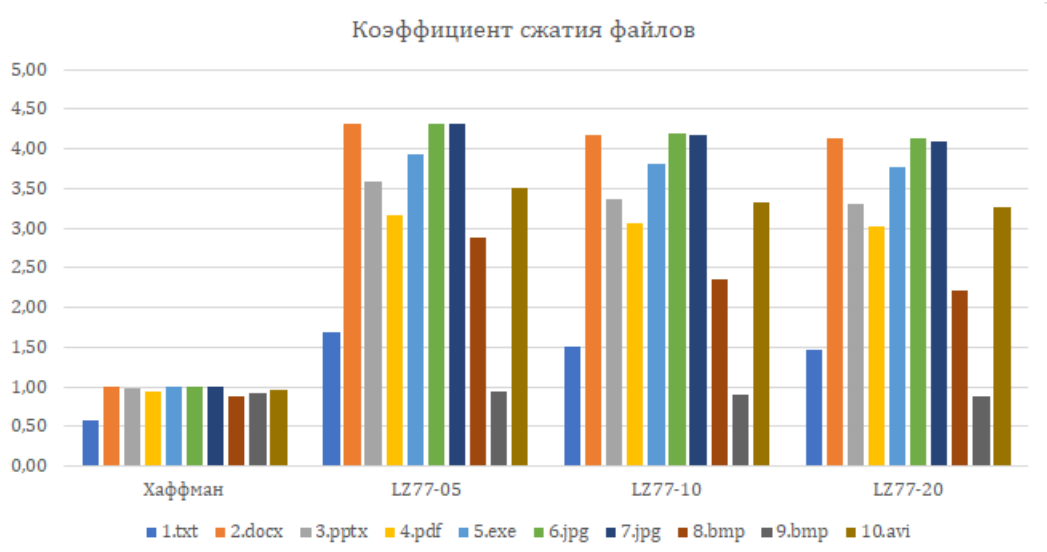

Рисунок 1. Диаграмма эффективности сжатия разных алгоритмов на разных файлах

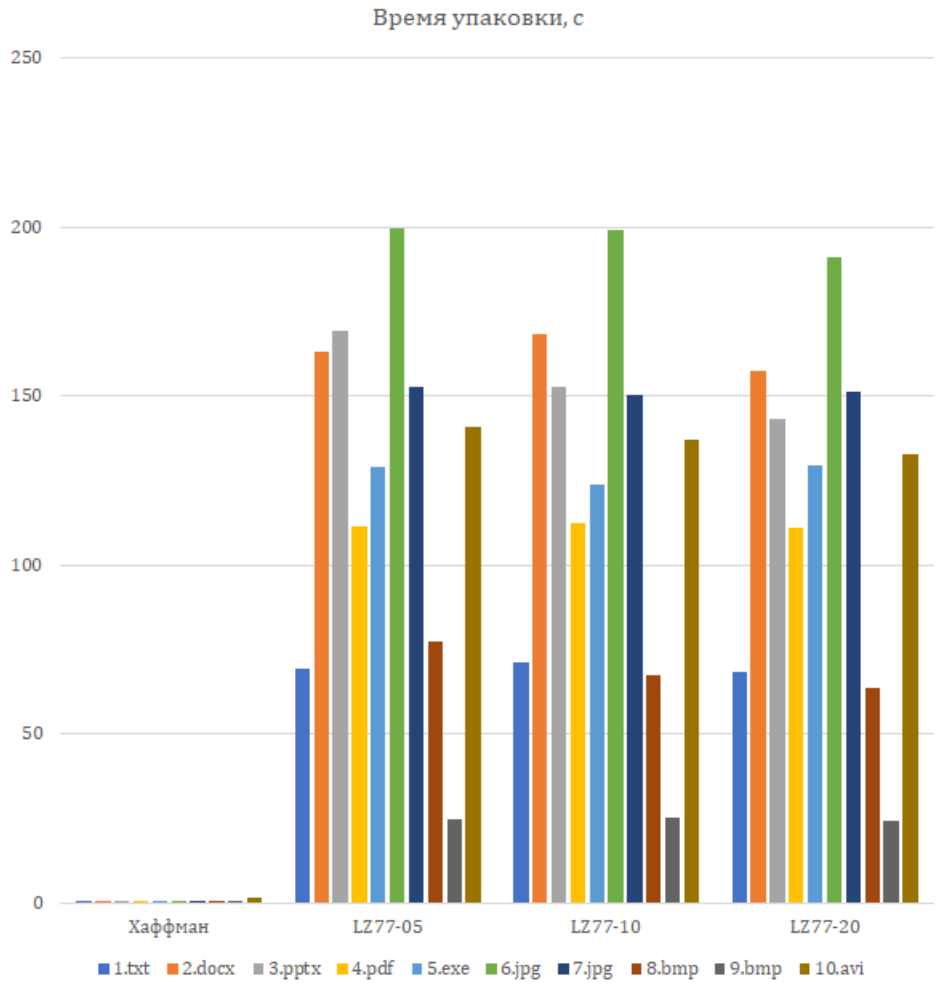

Рисунок 2. Диаграмма времени упаковки каждого файла для каждого алгоритма

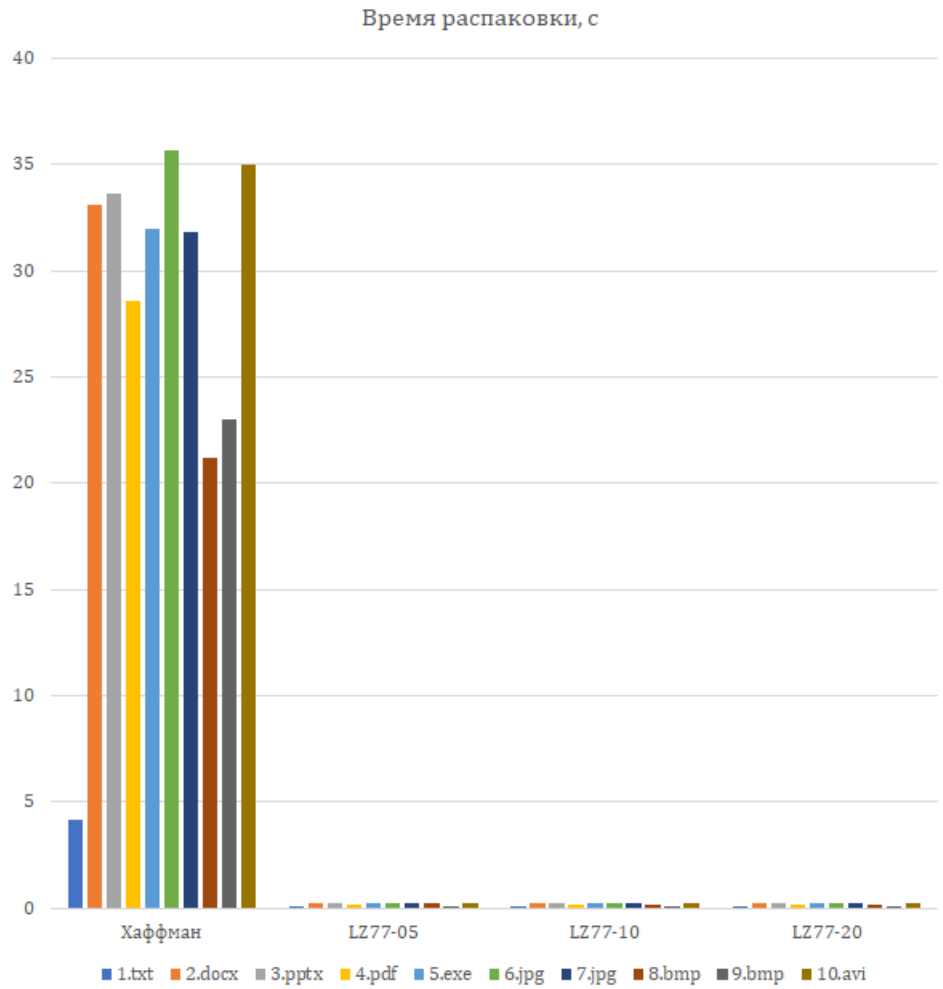

Рисунок 3. Диаграмма времени распаковки каждого файла для каждого алгоритма

6 Сравнительный анализ алгоритмов

Для большинства файлов использование алгоритмов сжатия LZ77 оказалось неэффективным, так как размер архивированного файла зачастую был больше исходного, что связано с небольшим количеством повторов в байткодах архивируемых файлов.

Использование алгоритма Хаффмана давало лучший коэффициент сжатия файлов. Стоит заметить, что на небольших файлах алгоритм Хаффмана также давал увеличение в размере файла (было выяснено на этапе тестирования алгоритмов).

По скорости сжатия наилучший результат имел алгоритм Хаффмана, в то время как архивирование с помощью алгоритма LZ77 происходило значительно дольше. При распаковке файлов наблюдалась ровно противоположная ситуация: при использовании алгоритмов LZ77 разархивирование файлов происходило гораздо быстрее, чем при использовании алгоритма Хаффмана.

7 Заключение

Из всех исследованных алгоритмов на использованных входных данных в подавляющем большинстве случаев по степени сжатия лучше оказывался алгоритм Хаффмана. Алгоритмы LZ77 с различными размерами скользящего окна и буфера истории показывали гораздо более худший результат. Несмотря на то, что суммарное время упаковки и распаковки алгоритмами LZ77 меньше, чем алгоритмом Хаффмана, это не дает им преимущества из-за их плохой эффективности в чистом виде.