Частота использования

В английском языке некоторые буквы употребляются намного чаще других. Например, буквы E и T используются очень часто, в то время как буквы X и Q встречаются довольно редко. В Таблице 1 приведены нормированные частоты использования всех букв, используемых в версии Библии Кинга Джеймса (King James). Можно видеть, что буква E, используемая чаще всех других букв, встречается примерно в пять раз чаще, чем буквы U и M, стоящие в середине списка, и примерно в 400 раз чаще, чем буква Q, которая завершает список.

Таблица 1.

Частота использования букв в Библии Кинга Джеймса (King James)

Частота	Буква	Частота	Буква	Частота	Буква
0,1272	E	0,0489	D	0,0151	B
0,0981	T	0,0401	L	0,0133	P
0,0875	H	0,0258	F	0,0094	V
0,0852	A	0,0257	U	0,0069	K
0,0750	O	0,0247	M	0,0027	J
0,0695	N	0,0202	W	0,0009	Z
0,0598	I	0,0181	Y	0,0004	X
0,0587	S	0,0170	G	0,0003	Q
0,0525	R	0,0169	C		

Частота употребления букв учитывалась при разработке кода Морзе (Таблица 2), в котором чаще встречаемым буквам присвоены более короткие коды. Так, буквам E и T соответствуют 1-символьные коды, в то время как буквам, используемым реже, присвоены 4-символьные коды.

Таблица 2.

Код Морзе.

Код Морзе	Буква	Код Морзе	Буква	Код Морзе	Буква
.	E	-..	D	-...	B
-	T	.-..	L	.--.	P
....	H	..-.	F	...-	V
.-	A	..-	U	-.-	K
---	O	--	M	.---	J
-.	N	.--	W	--..	Z
..	I	-.--	Y	-..-	X
...	S	--.	G	--.-	Q
.-.	R	-.-.	C		

В мире компьютеров наборы символов в латинских алфавитах почти всегда представляют путем применения методов кодирования с фиксированной длиной кода (fixed-length encoding methods), таких как ASCII или EBCDIC, которые не принимают в расчет частоту использования. Каждый символ кодируется постоянным числом битов, независимо от того, как часто он встречается. Это весьма эффективно для достижения наивысшей скорости вычислений. Однако там, где размер данных играет важную роль, есть смысл использовать коды переменной длины.

Алгоритм Шеннона — Фано — один из первых алгоритмов сжатия, который впервые сформулировали американские учёные Шеннон и Фано. Алгоритм использует коды переменной длины: часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся — кодом большей длины. Коды Шеннона — Фано префиксные, то есть никакое кодовое слово не является префиксом любого другого. Это свойство позволяет однозначно декодировать любую последовательность кодовых слов.

Основные этапы:

1. Символы первичного алфавита m_1 выписывают в порядке убывания вероятностей.
2. Символы полученного алфавита делят на две части, суммарные вероятности символов которых максимально близки друг другу.
3. В префиксном коде для первой части алфавита присваивается двоичная цифра «0», второй части — «1».
4. Полученные части рекурсивно делятся и их частям назначаются соответствующие двоичные цифры в префиксном коде.

Когда размер подалфавита становится равен нулю или единице, то дальнейшего удлинения префиксного кода для соответствующих ему символов первичного алфавита не происходит, таким образом, алгоритм присваивает различным символам префиксные коды разной длины. На шаге деления алфавита существует неоднозначность, так как разность суммарных вероятностей может быть одинакова для двух вариантов разделения (учитывая, что все символы первичного алфавита имеют вероятность больше нуля).

Код Шеннона — Фано строится с помощью дерева. Построение этого дерева начинается от корня. Всё множество кодируемых элементов соответствует корню дерева (вершине первого уровня). Оно разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Эти подмножества соответствуют двум вершинам второго уровня, которые соединяются с корнем. Далее каждое из этих подмножеств разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Им соответствуют вершины третьего уровня. Если подмножество содержит единственный элемент, то ему соответствует концевая вершина кодового дерева; такое подмножество разбиению не подлежит. Подобным образом поступаем до тех пор, пока не получим все концевые вершины. Ветви кодового дерева размечаем символами 1 и 0.

Помните, что при работе с кодами Хаффмана оперируют битовыми строками. Код Хаффмана состоит из переменного числа бит. Иными словами, это строка переменной длины, которая может состоять из нулей или единиц.

При построении кода Шеннона — Фано разбиение множества элементов может быть произведено, несколькими способами. Выбор разбиения на уровне n может ухудшить варианты разбиения на следующем уровне (n + 1) и привести к неоптимальности кода в целом. Другими словами, оптимальное поведение на каждом шаге пути ещё не гарантирует оптимальности всей совокупности действий. Поэтому код Шеннона — Фано не является оптимальным в общем смысле, хотя и дает оптимальные результаты при некоторых распределениях вероятностей. Для одного и того же распределения вероятностей можно построить, вообще говоря, несколько кодов Шеннона — Фано, и все они могут дать различные результаты. Если построить все возможные коды Шеннона — Фано для данного распределения вероятностей, то среди них будут находиться и оптимальные коды.

Чтобы продемонстрировать кодирование Шеннона — Фано на примере, воспользуемся следующим палиндромом (палиндромом называется строка, читаемая одинаково в обоих направлениях). Палиндром состоит из восьми разных символов с частотами употребления, приведенными в Таблице 3.

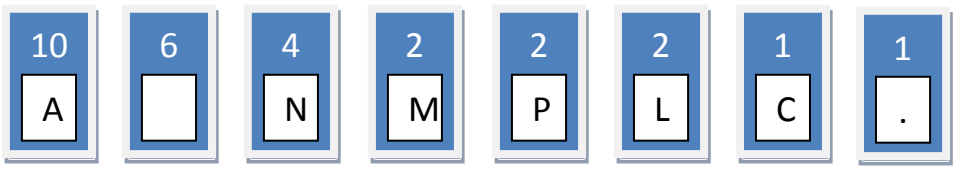
A MAN A PLAN A CANAL PANAMA.

Таблица 3.

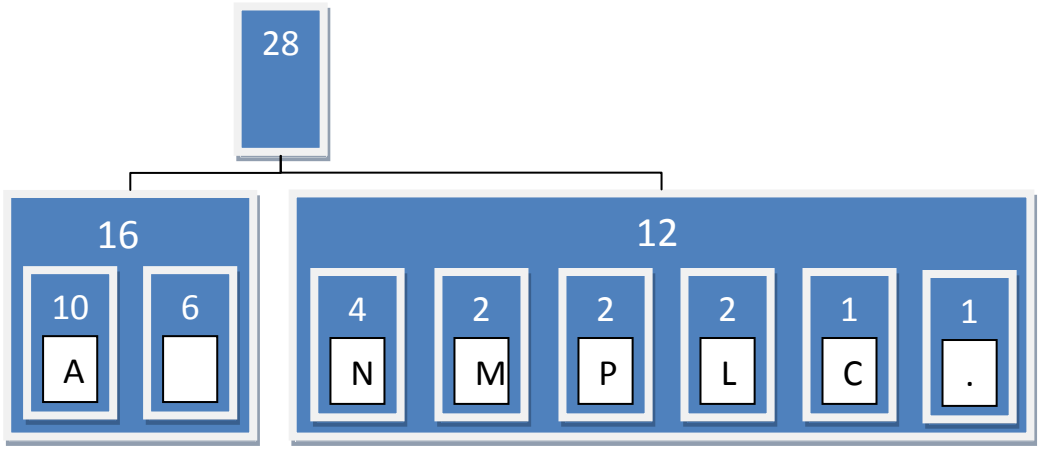
Частоты употребления символов в палиндроме.

Значение	Частота	Значение	Частота
A	10	N	4
C	1	P	2
L	2	Пробел	6
M	2	. (точка)	1

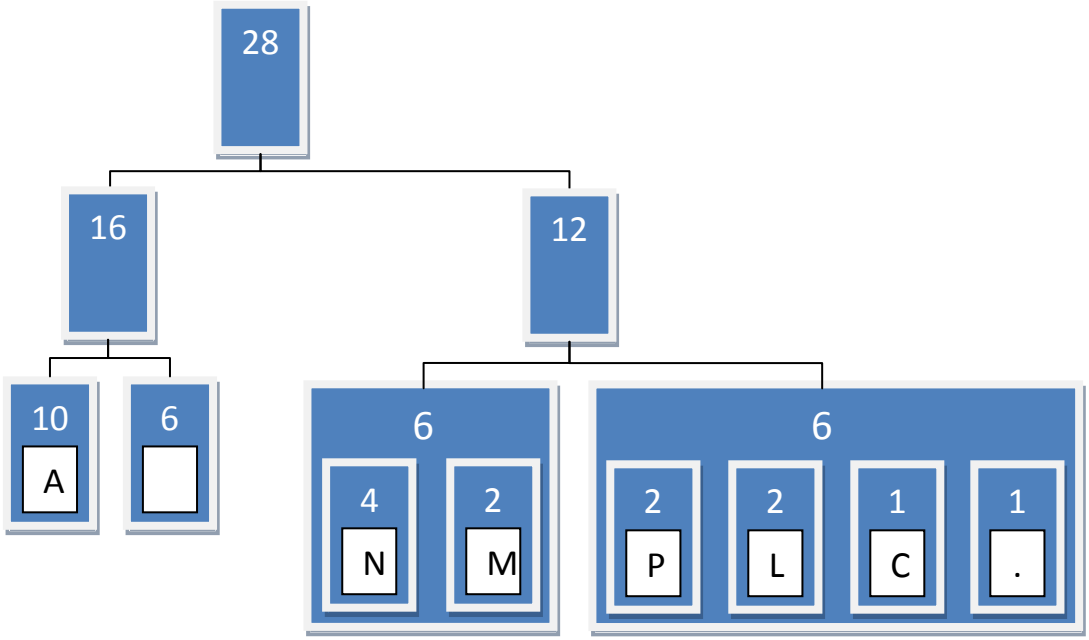
Перед построением кода необходимо определиться с порядком следования символов в последовательности, поскольку от него зависит оптимальность результата. В большинстве случаев результат близкий к оптимальному можно получить, если упорядочить символы по убыванию вероятности их появления, хотя это не гарантирует оптимальность.



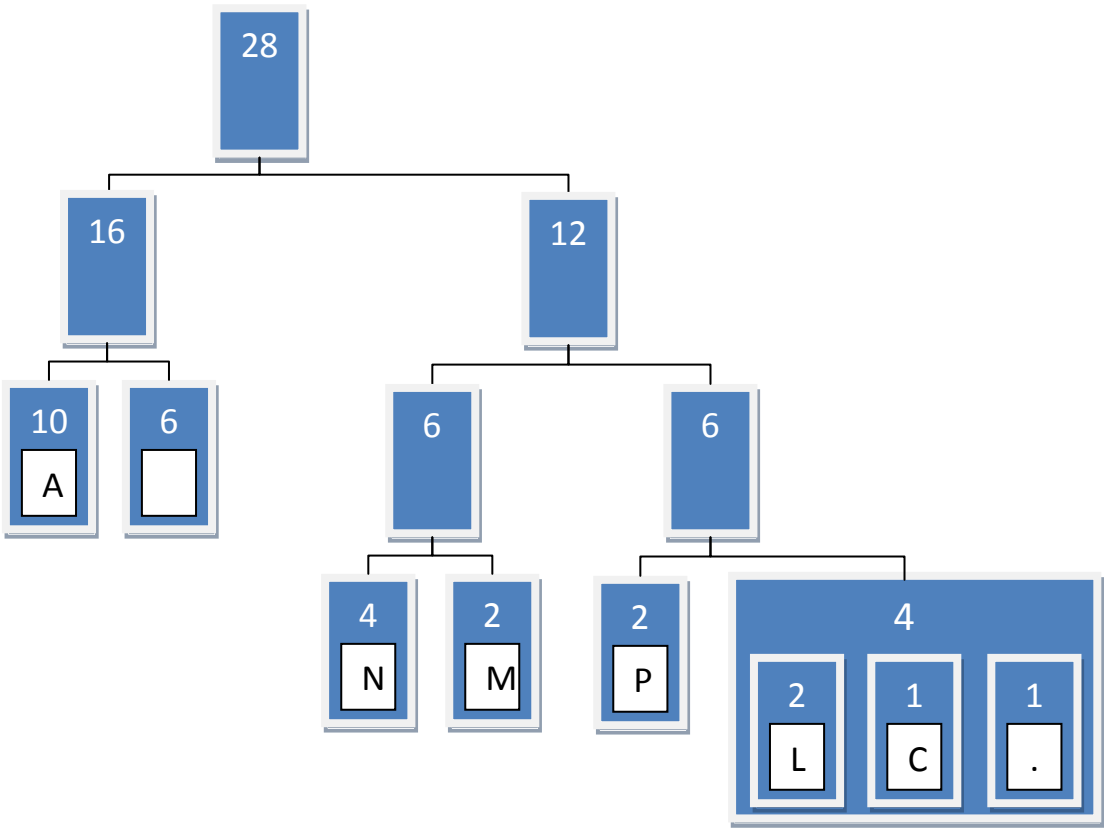
На следующем этапе необходимо произвести разбиение списка на два подсписка, имеющих примерно одинаковые веса. При разбиении списка перестановка элементов не допускается.



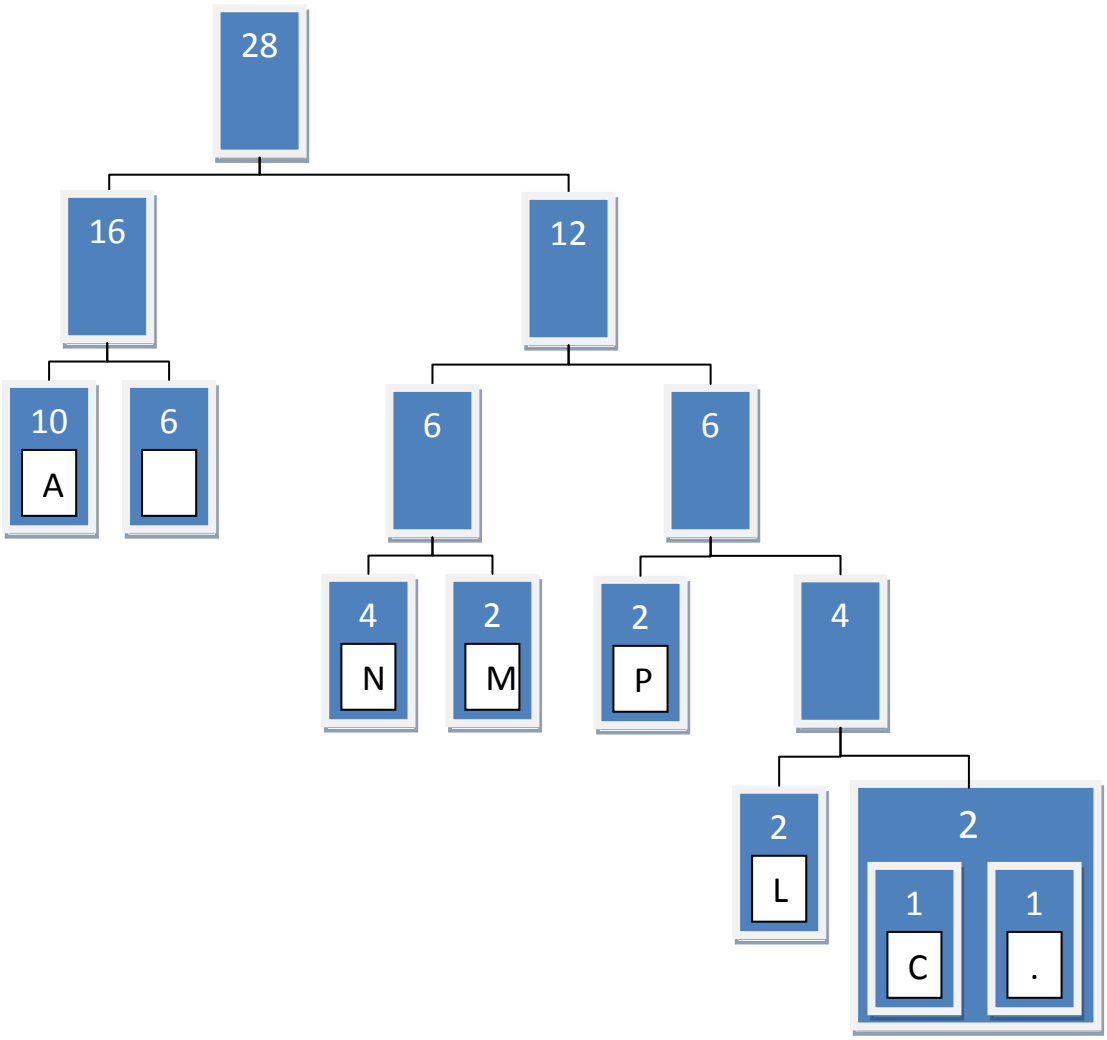
Продолжаем разбиение каждого из подсписков до тех пор, пока подсписки не выродятся в листья дерева.

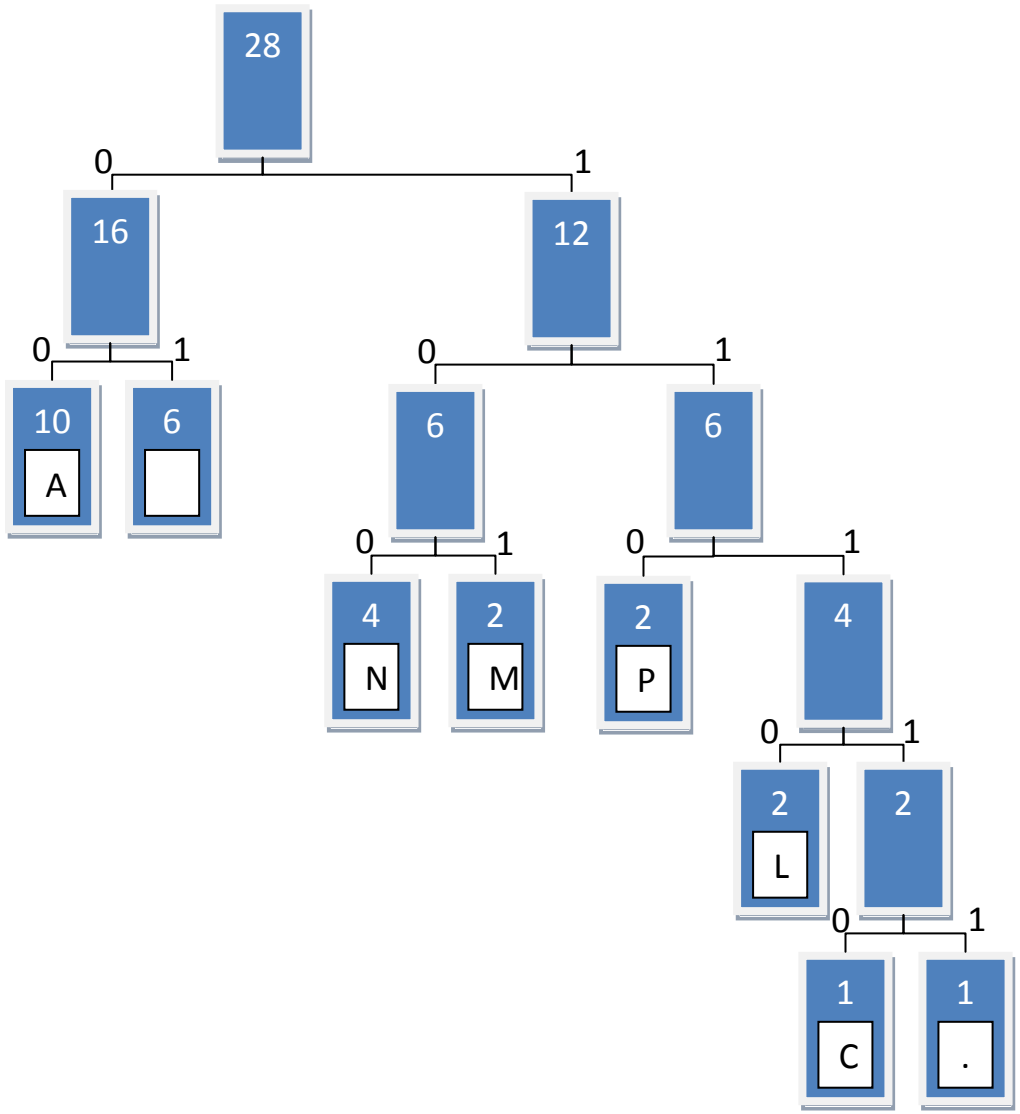


В полученном дереве может вызвать трудности разбиение списка «**PLC.**», поскольку есть два равноценных варианта разбиения «**P**»-«**LC.**» либо «**PL**»-«**C.**» имеющие веса 2:4 и 4:2 соответственно. Алгоритм Шенона-Фано не регламентирует, какой из списков выбирать в данном случае. Тем не менее, если в подобной ситуации руководствоваться правилом, согласно которому выбор нужно производить в пользу варианта разбиения, при котором вес левого подсписка должен быть меньше веса правого подсписка, больше вероятность получить оптимальный код. Однако это может привести к увеличению длины кода (максимальной длины которую)



Согласно предыдущему правилу, при разбиении списка «**PLC.**» на два подсписка был выбран вариант списков «**P**»-«**LC.**»


28
16
12
10
A
6
6
6
4
N
2
M
2
P
4
2
L
2
1
C
1
.



Код Шенона-Фано для каждого кодируемого символа создается путем присоединения к коду 0 или 1, при прохождении по пути от корня к кодируемому символу. В Таблице 4 приведены коды Хаффмана, полученные для каждого символа. Используя коды Хаффмана, показанные в Таблице 4, палиндром можно закодировать с помощью 74 бит, а если бы применялись коды постоянной длины, то это потребовало бы использования 3 бита (минимально необходимое число бит, для кодирования 8 символов) для кодирования каждого символа и 84 (3x28) бита для кодирования всей строки. Хотя в данном примере получено скромное сжатие на 12%, не забывайте, что мы работали с достаточно коротким текстом и между частотами символов нет большой разницы. Вы не получите разброса частот порядка 300:1 в текстовой строке из 28 символов.

Обратите внимание, что в Таблице 4 ни один из кодов не является префиксом к любому другому коду. Например, букве *N* присвоен код 100, и ни один из кодов в таблицы не начинается с битовой строки 100. Данное положение верно и для всех остальных кодов. Это очень важное свойство, поскольку без него оказалось бы невозможным декодировать строку, закодированную по методу Шенона-Фано.

Таблица 4.

Коды Шенона-Фано для символов палиндрома

Значение	Код *Шенона-Фано*	Длина кода	Частота использования символов	Использовано бит
A	00	2	10	20
C	11110	5	1	5
L	1110	4	2	8
M	101	3	2	6
N	100	3	4	12
P	110	3	2	6
Пробел	01	2	6	12
. (точка)	11111	5	1	5
			Всего:	74

Кодирование Хаффмана

Наиболее известный метод генерирования кодов переменной длины для символов, на основе частоты их употребления, называется *кодированием Хаффмана* (Huffman coding). Этот метод предложен Хаффманом в 1952 году. Процедура формирования кодов Хаффмана для набора значений, на основе частот их употребления, достаточно проста. Она включает создание двоичного дерева, содержащего символы, начиная снизу вверх, причем символы, которые встречаются реже всего, находятся дальше всех от корня. Сначала создается пул (pool), который содержит или значения, или узлы дерева. Первоначально этот пул содержит все значения и не содержит узлов. Приводимая ниже процедура повторяется до тех пор, пока пул не будет содержать один узел дерева, и в нем не будет ни одного символа.

1. Найдите два значения или узла дерева с наименьшей частотой появления и удалите их из пула. Если есть несколько элементов с наименьшей частотой использования, связь можно разорвать с помощью произвольного выбора. (Выбор, выполняемый для разрыва связи, влияет только на коды, генерируемые для кодируемых значений и не влияют на степень сжатия.)
2. Создайте новый узел дерева и сделайте элементы из предыдущего шага его двумя ветвями.
3. Присвойте новому узлу дерева частоту равную сумме частот отходящих от него ветвей.
4. Добавьте вновь созданный узел в пул.

После того как все значения будут объединены в одно дерево, для каждого узла дерева назначается значение 0 одной его ветви и значение 1 другой ветви узла. Код Хаффмана для каждого кодируемого значения определяется проходом по пути от корня дерева до данного значения, и добавлением в общий код кода каждой ветви, встречаемой по мере продвижения. Помните, что при работе с кодами Хаффмана

оперируют битовыми строками. Код Хаффмана состоит из переменного числа бит. Иными словами, это строка переменной длины, которая может состоять из нулей или единиц. Множество кодируемых значений и связанных с ними кодов образуют так называемую таблицу Хаффмана.

Чтобы продемонстрировать кодирование Хаффмана на примере, воспользуемся следующим палиндромом (палиндромом называется строка, читаемая одинаково в обоих направлениях). Палиндром состоит из восьми разных символов с частотами употребления, приведенными в Таблице 5.

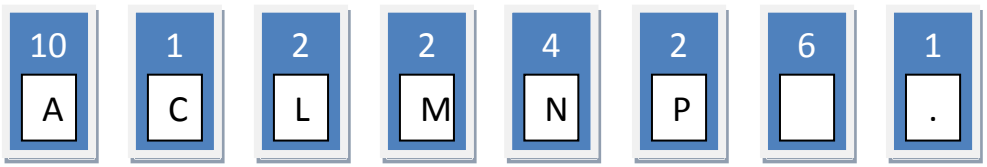
A MAN A PLAN A CANAL PANAMA.

Таблица 5.

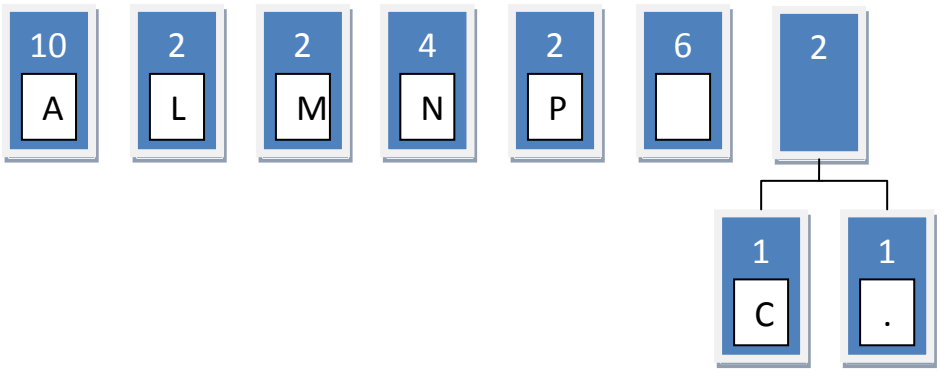
Частоты употребления символов в палиндроме.

Значение	Частота	Значение	Частота
A	10	N	4
C	1	P	2
L	2	Пробел	6
M	2	. (точка)	1

Как видно из Таблицы 3, символы «**C**» и точки («**.**») встречаются реже всех. Чтобы начать процесс кодирования Хаффмана, возьмем эти символы для создания узла дерева.



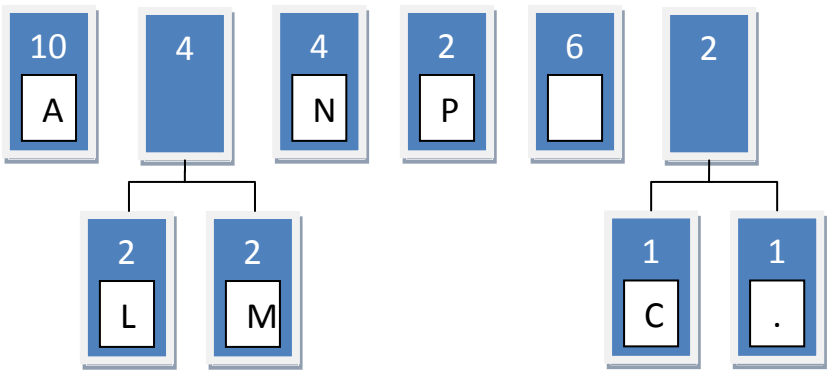
Назначим этому узлу значение частоты равное сумме частот отходящих от узла ветвей.



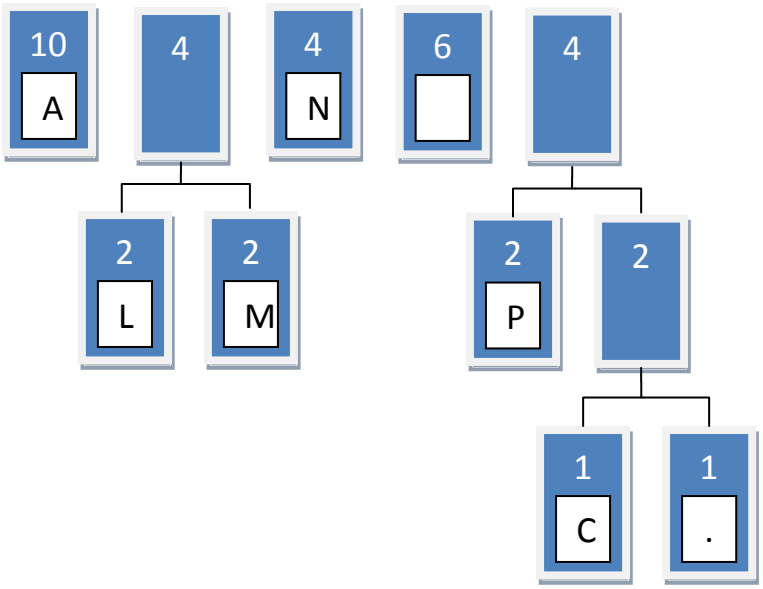
Четыре из оставшихся в пуле элементов связаны с низшей частотой равной 2. Когда возникает несколько элементов с одинаковой частотой встает проблема выбора элементов, для объединения выбор повлияет на структуру дерева, но как ни странно не повлияет на эффективность сжатия. Для удобства введем правило: Если в пуле

присутствуют более двух элементов, или узлов, с одинаковой частотой выберем для объединения элементы, или узлы, наименьшей длинны. В данном случае это правило ограничит выбор символами «**L**» «**M**» «**P**», исключив из рассмотрения узел «**C**» «**.**», в дальнейшем это позволит сократить наибольшую длину кода и получить более сбалансированное дерево. Выбирая между символами «**L**», «**M**» и «**P**» объединим первые два символа в произвольном порядке (это никак не скажется на структуре дерева).

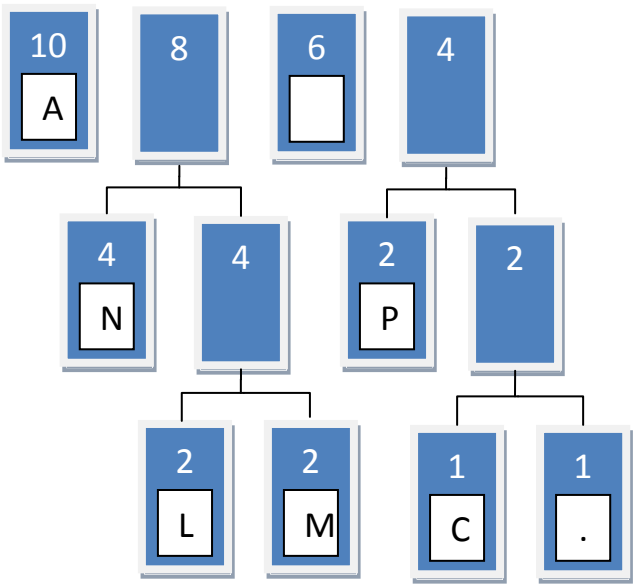
Берем Символ «**L**» и «**M**», соединяем их друг с другом, чтобы создать новый узел дерева с суммарной частотой равной 4.



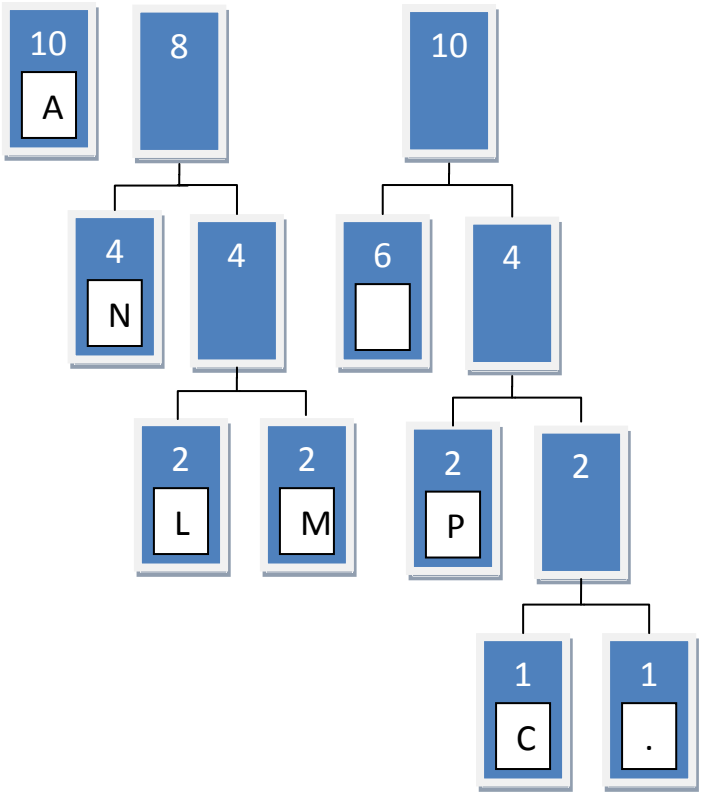
Теперь из оставшихся символов наименьшая частота встречается у буквы «**P**» и узла «**C**» - «**.**» объединим их.



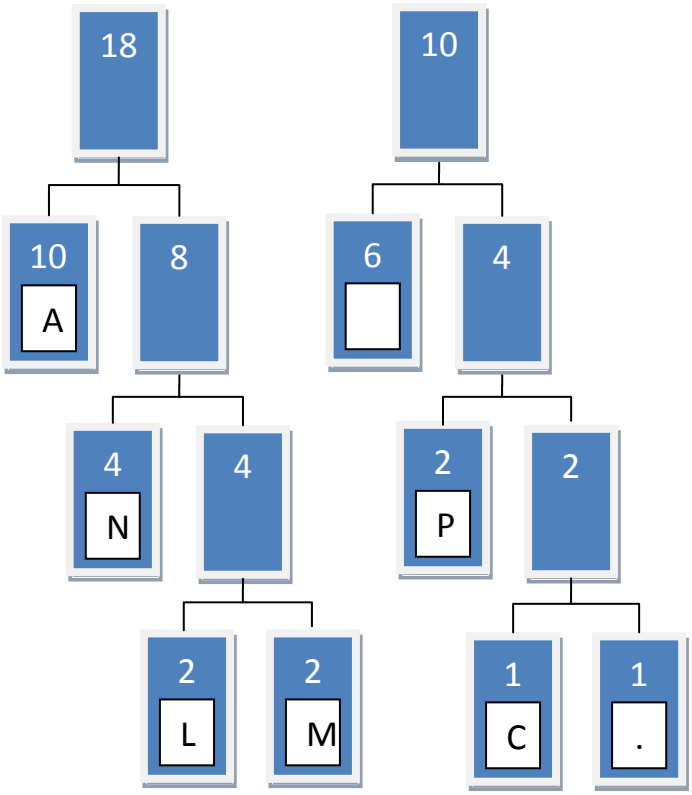
Теперь есть два узла дерева и буква «**N**», связанные с наименьшей частотой равной 4. Мы выбираем букву «**N**» и присоединяем ее к дереву.



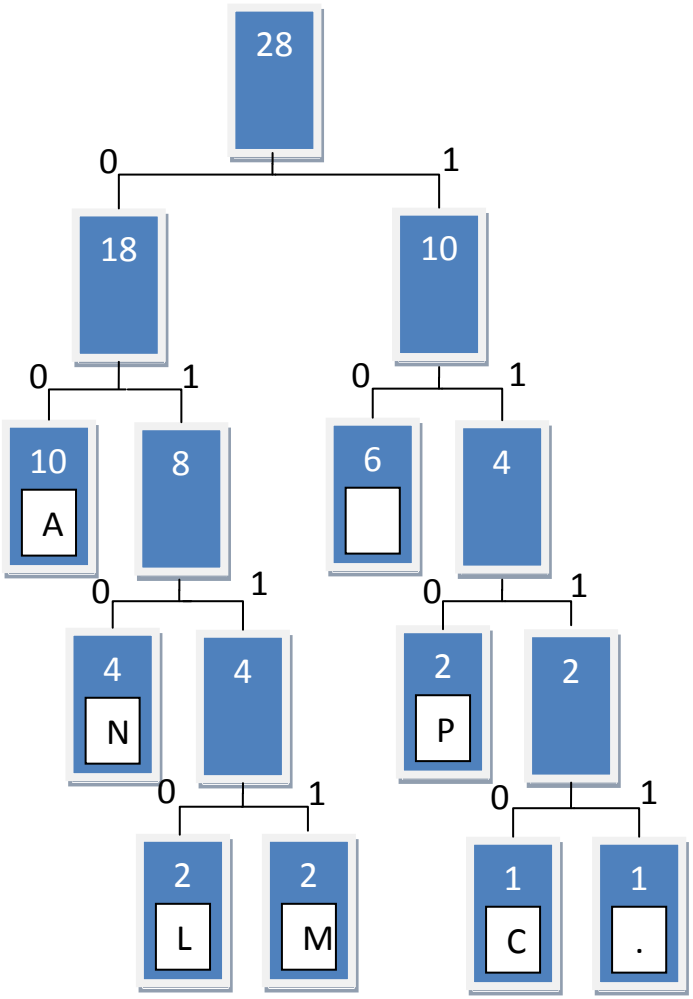
Из оставшихся элементов наименьшие частоты встречаются у символа пробела и узла дерева «**P**» - «**C**» - «**M**», поэтому они соединяются в одно дерево.


10
A
8
10
4
N
4
6
4
2
L
2
M
2
P
2
1
C
1
.

Теперь остались два узла дерева и буква «**A**». Мы произвольно решаем соединить букву с наиболее коротким узлом.



Наконец, соединяем два узла, чтобы завершить построение дерева, а затем помечаем каждую левую ветвь 0 (нулем), а каждую правую ветвь 1 (единицей).



Код Хаффмана для каждого кодируемого символа создается путем присоединения к коду 0 или 1, при прохождении по пути от корня к кодируемому символу. В Таблице 6 приведены коды Хаффмана, полученные для каждого символа. Используя коды Хаффмана, показанные в Таблице 4, палиндром можно закодировать с помощью 74 бит, а если бы применялись коды постоянной длины, то это потребовало бы использования 3 бита (минимально необходимое число бит, для кодирования 8 символов) для кодирования каждого символа и 84 (3x28) бита для кодирования всей строки. Хотя в данном примере получено скромное сжатие на 12%, не забывайте, что мы работали с достаточно коротким текстом и между частотами символов нет большой разницы. Вы не получите разброса частот порядка 300:1 в текстовой строке из 28 символов.

Таблица 6.

Коды Хаффмана для символов палиндрома

Значение	Код Хаффмана	Длина кода	Частота использования символов	Использовано бит
A	00	2	10	20
C	1110	4	1	4
L	0110	4	2	8
M	0111	4	2	8
N	010	3	4	12
P	110	3	2	6
Пробел	10	2	6	12
. (точка)	1111	4	1	4
			Всего:	74

Обратите внимание, что в Таблице 4 ни один из кодов не является префиксом к любому другому коду. Например, букве *N* присвоен код 011, и ни один из кодов в таблицы не начинается с битовой строки 011. Данное положение верно и для всех остальных кодов. Это очень важное свойство, поскольку без него оказалось бы невозможным декодировать строку, закодированную по методу Хаффмана.

Если мы, интереса ради, попробуем объединять символы и деревья по следующему правилу: Если в пуле присутствуют более двух элементов, или узлов, с одинаковой частотой выберем для объединения элементы, или узлы, наибольшей длинны. Это приведет к построению другого дерева и другого кода Хаффмана для палиндрома. Дерево полученное во втором случае существенно хуже сбалансировано, чем в первом, это приводит к тому, что коды Хаффмана сильно различаются по длине. В первом случае наибольшая длина кода составляла 4 бита, а наименьшая 2. Во втором случае наибольшая длина кода составляет 5 бит, а наименьшая 1 бит. Тем не менее, размер закодированного палиндрома остался тем же (Таблица 7).

У любой последовательности данных может быть множество кодов Хафмана, позволяющих получить оптимальный результат. Тем не менее, если дерево более сбалансировано и длина кодов различается слабо, работать с ними будет проще.

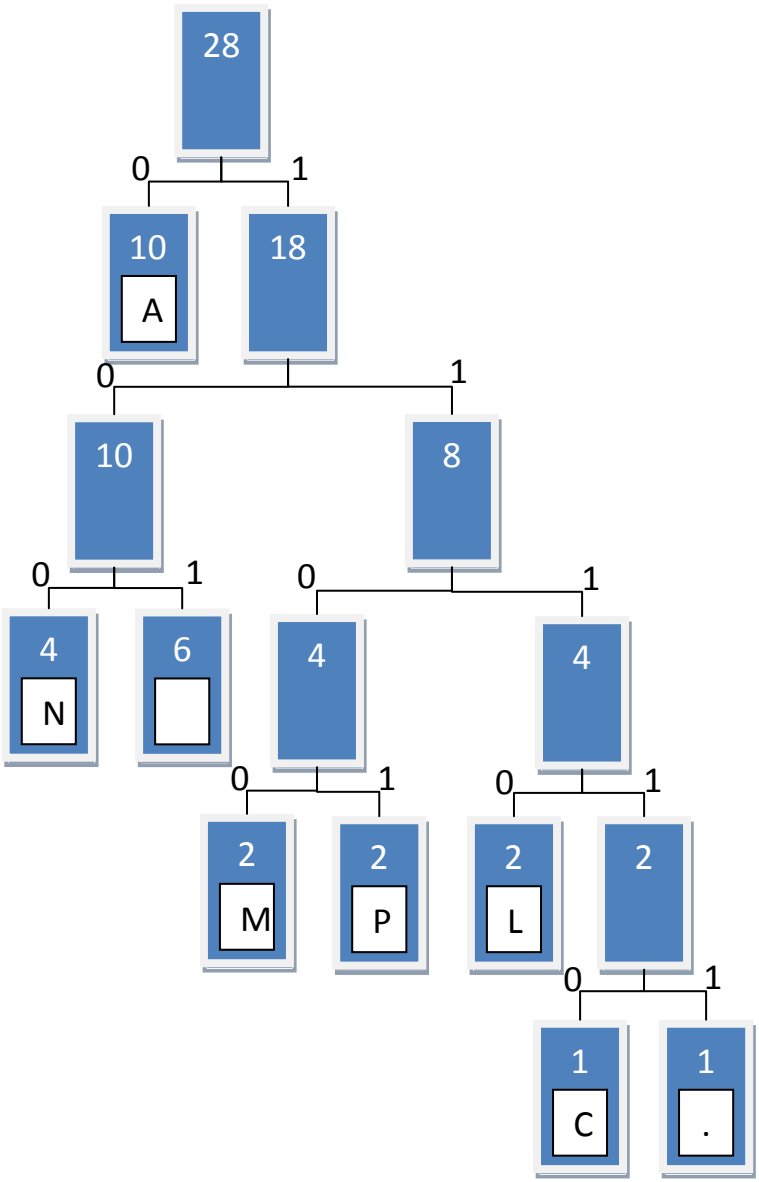


Таблица 7.

Коды Хаффмана для символов палиндрома

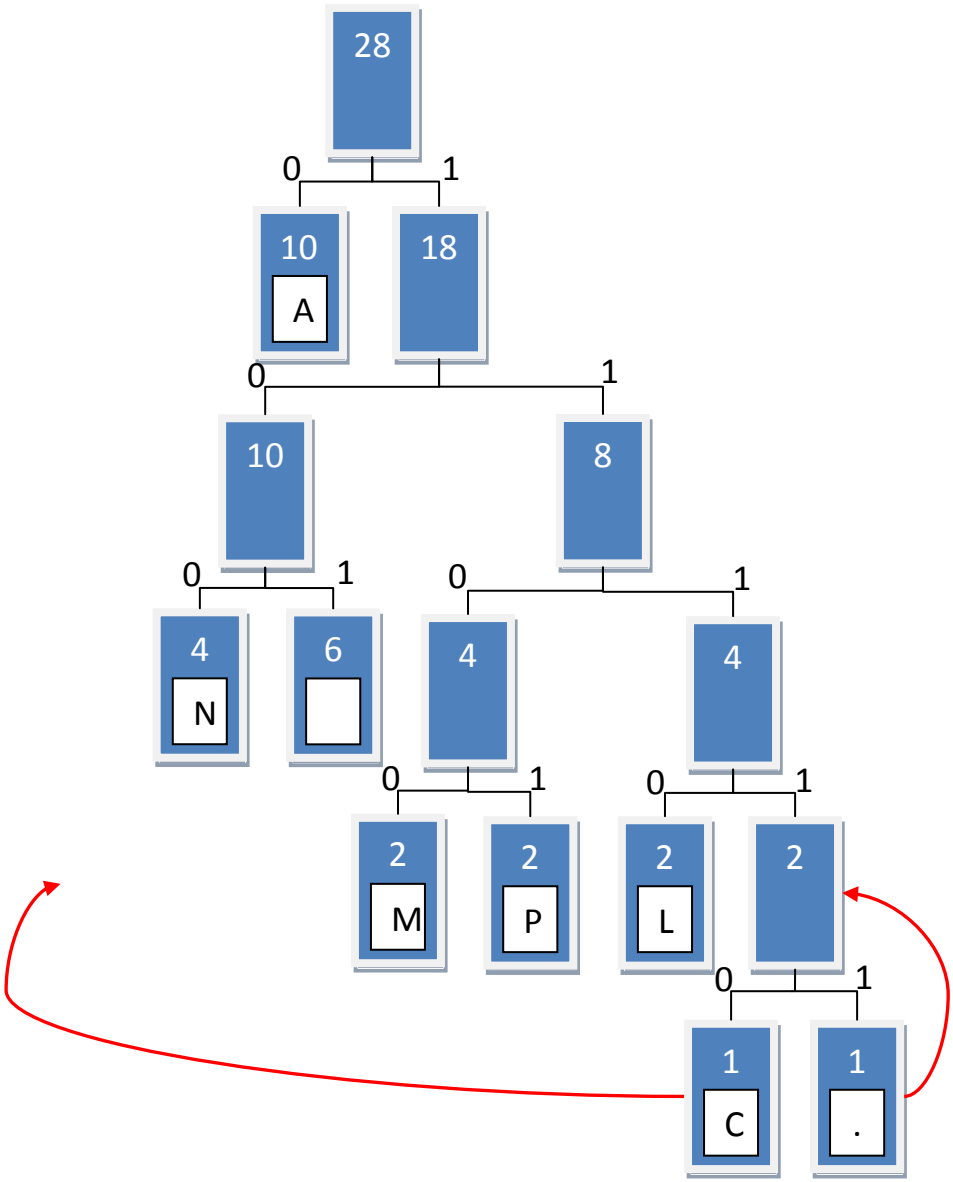
Значение	Код Хаффмана	Длина кода	Частота использования символов	Использовано бит
A	0	1	10	10
C	11110	5	1	5
L	1110	4	2	8
M	1100	4	2	8
N	000	3	4	12
P	1101	4	2	8
Пробел	001	3	6	18
. (точка)	11111	5	1	5
			Всего:	74

Уменьшение длины кода.

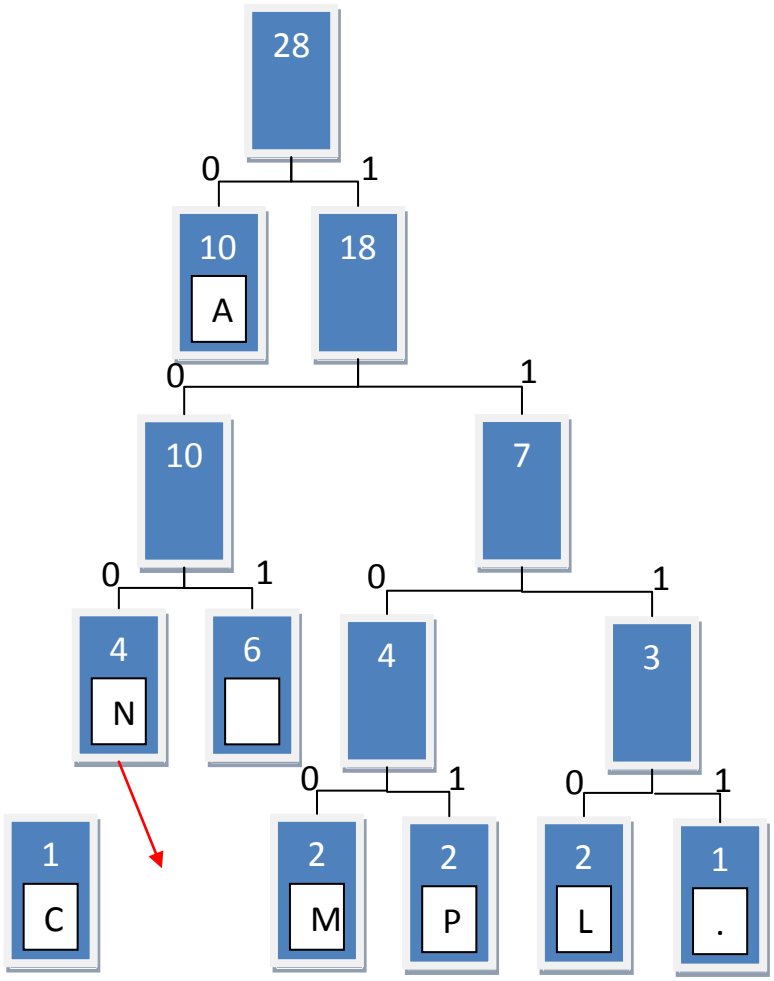
Алгоритмы сжатия, базирующиеся на кодах переменной длины, обладают существенным недостатком – в некоторых экстремальных случаях длина кода может превысить технические ограничения. Например, если сжимается обычный файл, при сжатии которого архиватор оперирует единицами данных, размером в байт (соответственно используется 256 кодовых элементов), максимальная длина кода может составить 255 бит. Хотя такая ситуация крайне маловероятна, она все-таки возможна. Оперировать кодами, максимальная длина которых неограниченна сложно, по этой причине на длину кода всегда накладывают ограничение. Поскольку, для наибольшей производительности длина кодового слова недолжна превышать величину операторного слова процессора, или должна быть кратна ему. Чаще всего выбирают ограничение на длину слова, равную 16, 32 или 64 битам.

При ограничении длинны кода необходимо перестроить дерево кодов Хаффмана, таким образом, чтобы максимальная длина ветви дерева не превосходила поставленное ограничение. Полученное после перестройки дерево уже не будет оптимальным деревом Хаффмана. Тем не может быть построено оптимальное дерево для данной длины ключа. Однако алгоритм построения оптимального дерева Хаффмана под код заданной длины слишком сложен и его рассмотрение не входит в данный курс. На практике, в большинстве случаев нет необходимости строить именно оптимальный код, поскольку разница в эффективности сжатия данных может быть несущественной, даже при использовании достаточно простого алгоритма балансировки дерева. Один из подобных алгоритмов показан ниже.

В предыдущем примере длина наибольшая длина слова составила 5 бит Таблица 5. Рассмотрим балансировку дерева до длины в 4 бита. При сокращении длины кода код перестанет быть оптимальным и эффективность сжатия уменьшится.



Для балансировки выберем лист дерева с наибольшей длинной ветви, в данном случае это будут листья «**P**» и «**.**» из поместим лист с наибольшим весом в общий узел (поскольку в данном случае листья имеют одинаковый вес выберем правый) второй лист сохраним во временной переменной.



Выберем лист полученного дерева, с наименьшим весом, длина ветви которого, как минимум на 2 короче максимальной длины ветви исходного дерева. В данном случае это будет лист «**N**». Заменим лист «**N**» на узел, к которому присоединим листья «**N**» и «**C**».

Проверим длину ветвей дерева, если она все еще превосходит заданную – повторим процедуру сокращения длины ветвей дерева.
Длина ветвей полученного дерева не превосходит заданную. Но если мы посмотрим на результат применения кодов к палиндрому (Таблица 8), то заметим, что размер увеличился на 2 бита, что не существенно, однако свидетельствует о неоптимальности дерева.

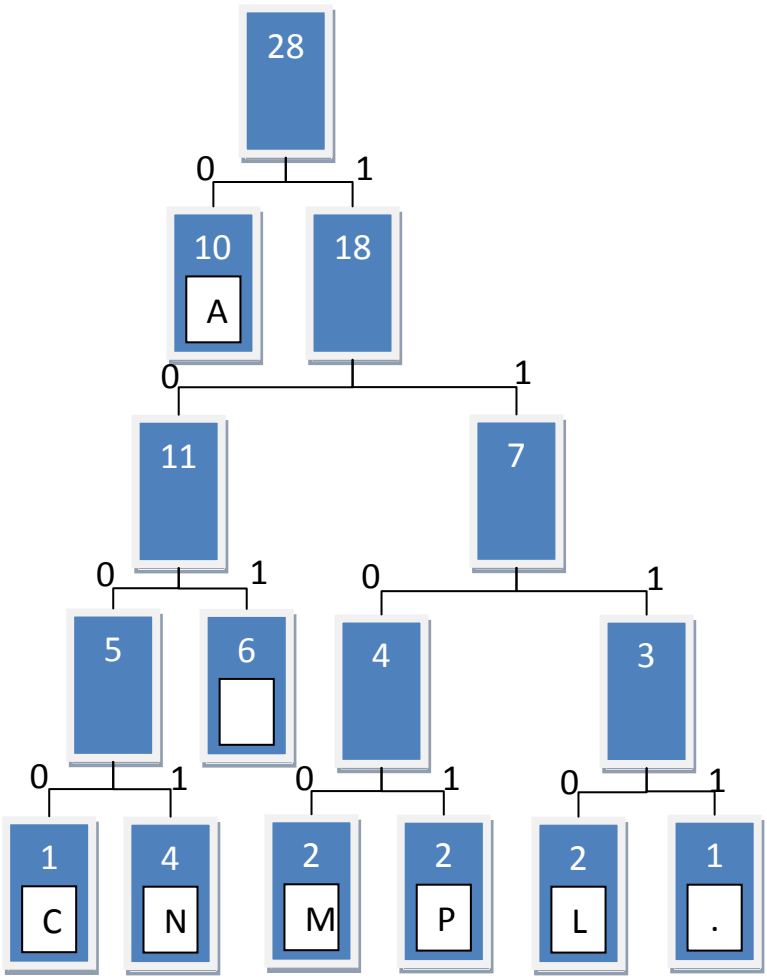


Таблица 8.
Коды Хаффмана для символов палиндрома

Значение	Код Хаффмана	Длина кода	Частота использования символов	Использовано бит
A	0	1	10	10
C	1000	4	1	4
L	1110	4	2	8
M	1100	4	2	8
N	1001	4	4	16
P	1101	4	2	8
Пробел	101	3	6	18
. (точка)	1111	4	1	4
			Всего:	76

Кодирование и декодирование файла

При использовании методов сжатия данных кодами переменной длинны, по полученному кодовому дереву формирует кодовая таблица. Кодовая таблица представляет собой динамический массив, либо список, из структур, хранящих исходный символ, полученный код символа, длину кода символа. Поскольку мы заранее не знаем длину кода, при задании структуры необходимо выбрать такой тип данных, под поле хранение кода, чтобы в нем гарантированно поместился код наибольшей длинны. В рамках данной лабораторной работы длина кода не превысит 32 бита, соответственно типа unsigned int будет достаточно, для хранения любого кода, который может возникнуть. Неиспользуемые биты кодового поля должны быть приравнены к 0.

Сам процесс кодирования, после построения кодовой таблицы осуществляется по следующему алгоритму:

1. Чтение символа из входного файла
2. Нахождение кодового слова, соответствующего входному символу и сохранение его во временную переменную, сохранение длины кодового слова в счетчик.
3. Сохранение младшего бита временной переменной в выходной поток.
4. Правый сдвиг временной переменной на 1 разряд.
5. Уменьшение счетчика на 1.
6. Проверка: Если счетчик > 0 переходим на 3
7. Пока не достигли конца входного файла переход на 1

Процесс декодирования осуществляется по следующему алгоритму:

1. Обнуление временной переменной.
2. Чтение бита из входного потока.
3. Прибавление считанного бита к временной переменной.
4. Проверка – присутствует ли в кодовой таблице символ с кодом, равным коду временной переменной, если нет, то выполняем левый сдвиг временной переменной на 1 разряд и переходим на 2.
5. Сохраняем в выходной файл байт, соответствующий кодовому символу.
6. Пока есть данные в потоке, переходим на 1

Поскольку ни один из кодов не является префиксом ни к одному другому коду, а неиспользуемые разряды заполнены 0, при чтении кодов нет необходимости отслеживать длину прочитанного кода.

Важный момент – перед декодированием кода необходимо иметь кодовую таблицу. Существует два подхода:

1. Можно сохранить кодовую таблицу в начале закодированного файла
2. Можно сохранить в начале закодированного файла частоты появления символов, либо их вероятности и на основе этих данных заново построить кодовую таблицу.

Первый вариант проще реализуется, но второй вариант в большинстве случаев занимает меньше файлового пространства, поэтому более распространен.

Задание на лабораторную работу

В процессе выполнения лабораторной работы необходимо подготовить текстовый файл формата txt, содержащий не менее 500 000 знаков. Для этого можно взять любимую книгу и сохранить ее в формате txt, либо переписать из электронной библиотеки книгу в формате txt. Поскольку в данном формате один знак в формате ASCII занимает 1 байт, размер файла должен быть не менее 500 000 байт.

Необходимо посимвольно считать текстовый файл, сформировать список входных символов, посчитав частоту появления каждого символа в файле, построить дерево кода, согласно заданию, сжать исходный файл, используя полученный код, сохранить результат в бинарный файл. Сохранить в новый текстовый файл таблицу, содержащую все символы исходного файла в алфавитном порядке с указанием частоты появления в тексте, вероятности появления в тексте, кода символа, длины кода символа. А также вывести аналогичную таблицу с символами, упорядоченными по вероятности появления в тексте. После этого необходимо восстановить исходный текстовый файл из закодированного бинарного файла, для проверки целостности исходного файла. Повторить сжатие и восстановление файла, сократив длину кода на 1, 2, 3 бита, каждый раз сохраняя таблицу с полученным кодом, упорядоченную по частоте использования символов. Оценить эффективность сжатия файла, сравнив размер исходного файла с размером сжатого файла для всех 4-х вариантов максимальных длин кодов.

Содержания отчета: Задание на работу, с указанием номера варианта и подробным текстовым описанием индивидуального задания. Название исходного файла с указанием причины выбора именно этого текстового файла. Таблицу частоты и вероятности появления символов, упорядоченную в алфавитном порядке, 4 таблицы, частоты и вероятности появления символов, с указанием кода, соответствующего символу и длины кода, для 4-х вариантов кодовых таблиц (исходная кодовая таблица, кодовая таблица с максимальной длиной кода уменьшенной на 1, 2, 3 бита). Сравнение эффективности сжатия исходного файла, для 4-х вариантов кодовых таблиц.

Номер варианта определяется по формуле: (последняя цифра номера группы * 5 + номер в журнале) mod 40, после чего берется задание из таблицы.

Сжатие данных \вариант	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Шенона-Фано	V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V	
Хафмана		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V		V
Сортировка по алфавиту в процессе создания списка	V	V			V	V			V	V			V	V			V	V			V	V			V	V			V	V			V	V			V	V		
Сортировка по алфавиту после создания списка			V	V			V	V			V	V			V	V			V	V			V	V			V	V			V	V			V	V		
Сохранение кодовой таблицы в формате: символ+частота появления символа	V			V			V			V			V			V			V			V			V			V			V			V			V			V
Сохранение кодовой таблицы в формате: символ+вероятность появления символа		V			V			V			V			V			V			V			V			V			V			V			V			V		
Сохранение кодовой таблицы в формате: символ+код символа			V			V			V			V			V			V			V			V			V			V			V			V			V	
Построение кодовой таблицы по частоте	V		V	V			V		V	V			V		V	V			V		V	V			V		V	V			V		V	V			V		V	V
Построение кодовой таблицы по вероятности		V			V	V		V			V	V		V			V	V		V			V	V		V			V	V		V			V	V		V		
Кодовая таблица: Динамический массив	V	V	V	V					V	V	V	V					V	V	V	V					V	V	V	V					V	V	V	V				
Кодовая таблица: Список					V	V	V	V					V	V	V	V					V	V	V	V					V	V	V	V					V	V	V	V