Статистические методы сжатия

Лекция 2

План

- Коды переменной длины
- Декодирование
- Кодирование Хаффмана
- Арифметическое сжатие

Коды переменной длины или выражайтесь ясно

- Дано

Символ	Вероятность появления	Code 1	Code 2
a1	0,49	1	1
a2	0,25	01	01
a3	0,25	010	000
a4	0,01	001	001

Минимальное число бит по теории информации 1,57

По Code1 среднее число бит
1x0.49+2x0.25+3x0.25+3x0.01=1.77

Коды переменной длины или выражайтесь ясно

- Закодируем строку

a1a3a2a1a3a3a4a2a1a1a2a2a1a1a3a1a1a2a3a1

Код строки

1|010|01|1|010|010|001|01|1|1|01|01|1|1|010|1|1|01|010|1

37 битов на 20 символов = 1,85 бит/символ

Декодируем

a1/a2?a3 – код двусмысленный

Свойство префикса

- Если некоторая последовательность би тов выбрана в качестве кода какого-то символа, то ни один код другого символа не должен иметь в начале эту последовательность

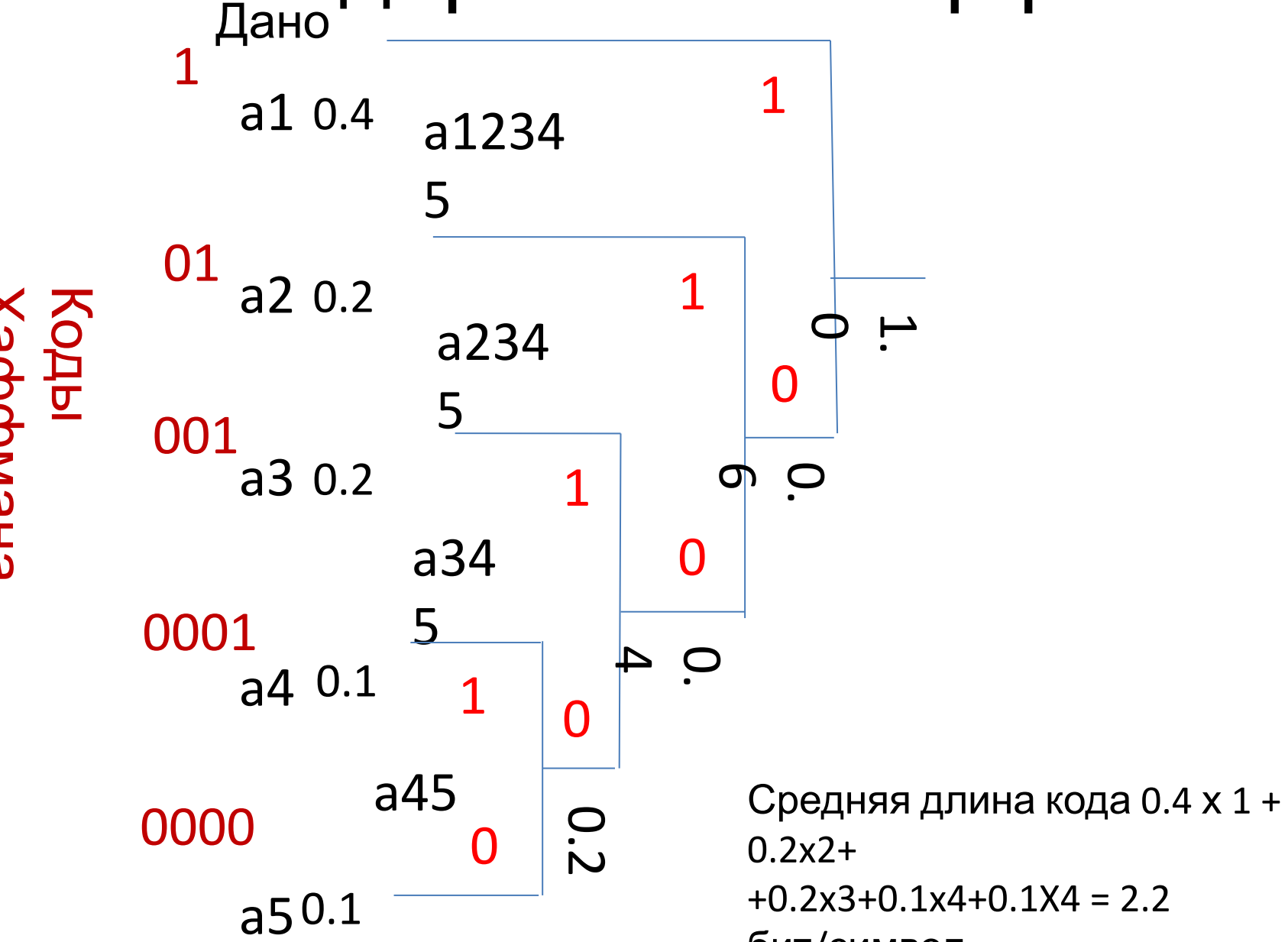
Правила назначения кодов переменной длины

- Следует назначать более короткие коды чаще встречающимся символам
- Коды должны удовлетворять свойству префикса

Декодирование

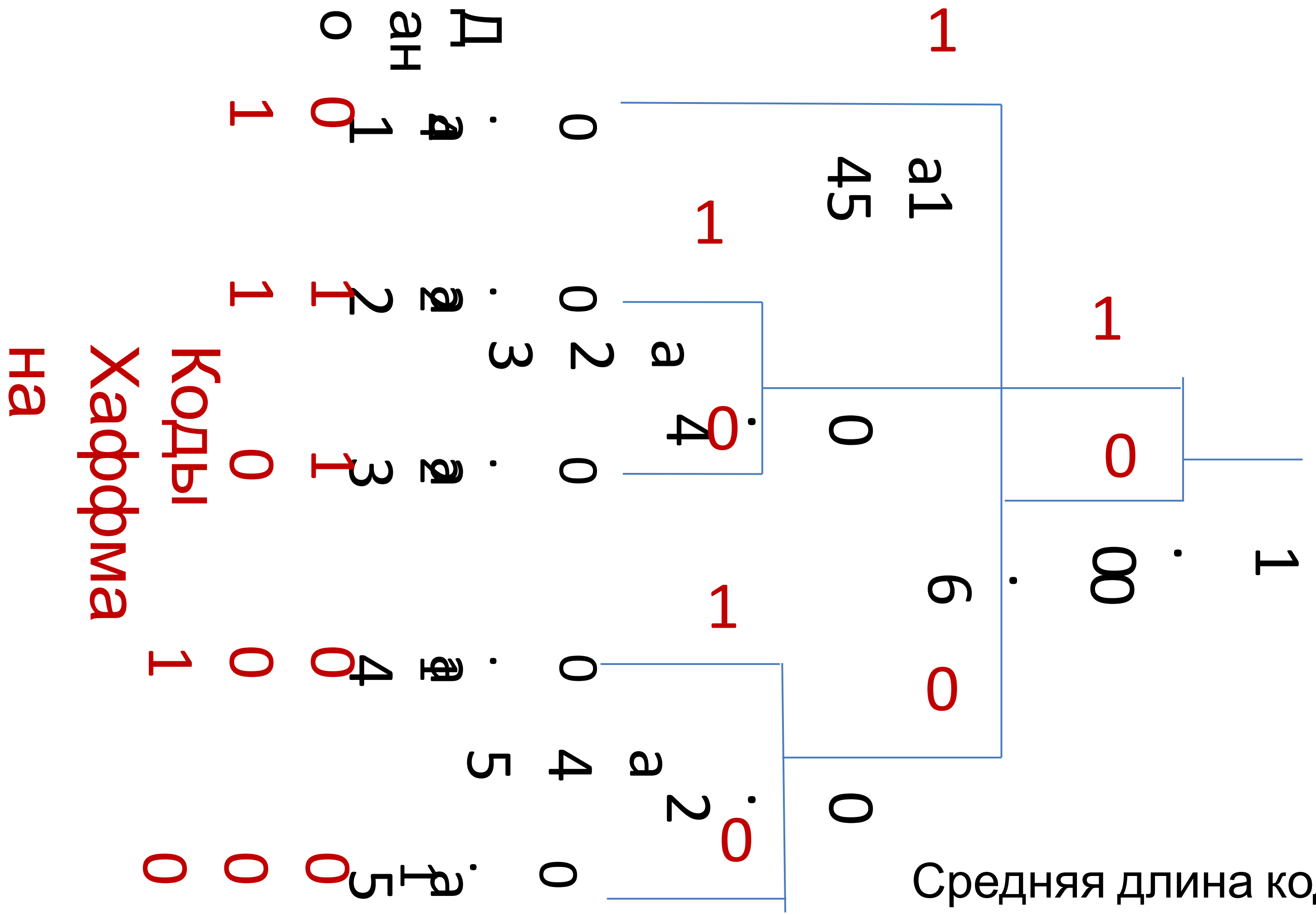
- Проблема – декодер должен знать префиксный код каждого символа
- Решения
  - Использовать набор стандартных префиксных кодов (факсимильная связь)
  - Кодер сканирует файл и передает информацию о статистических свойствах файла декодеру. Декодер подбирает префиксы
  - Кодер по мере работы над исходными данными улучшает исходный префиксный код. Декодер повторяет каждый шаг кодера

Кодирование Хаффмана



Средняя длина кода 0.4 x 1 +
0.2x2+
+0.2x3+0.1x4+0.1X4 = 2.2
бит/символ

Кодирование Хаффмана



Средняя длина кода 0.4 x 2 + 0.2x2+
+0.2x2+0.1x3+0.1X3 = 2.2

Выбор кода Хаффмана

- Наилучший код – с минимальной дисперсией
- Дисперсия кода 1

$$0.4\times(1-2.2)^2+0.2\times(2-2.2)^2+0.2\times(3-2.2)^2+$$
$$+0.1\times(4-2.2)^2+0.1\times(4-2.2)^2=1.36$$

- Дисперсия кода 2

$$0.4\times(2-2.2)^2+0.2\times(2-2.2)^2+0.2\times(2-2.2)^2+$$
$$+0.1\times(3-2.2)^2+0.1\times(3-2.2)^2=0.16$$

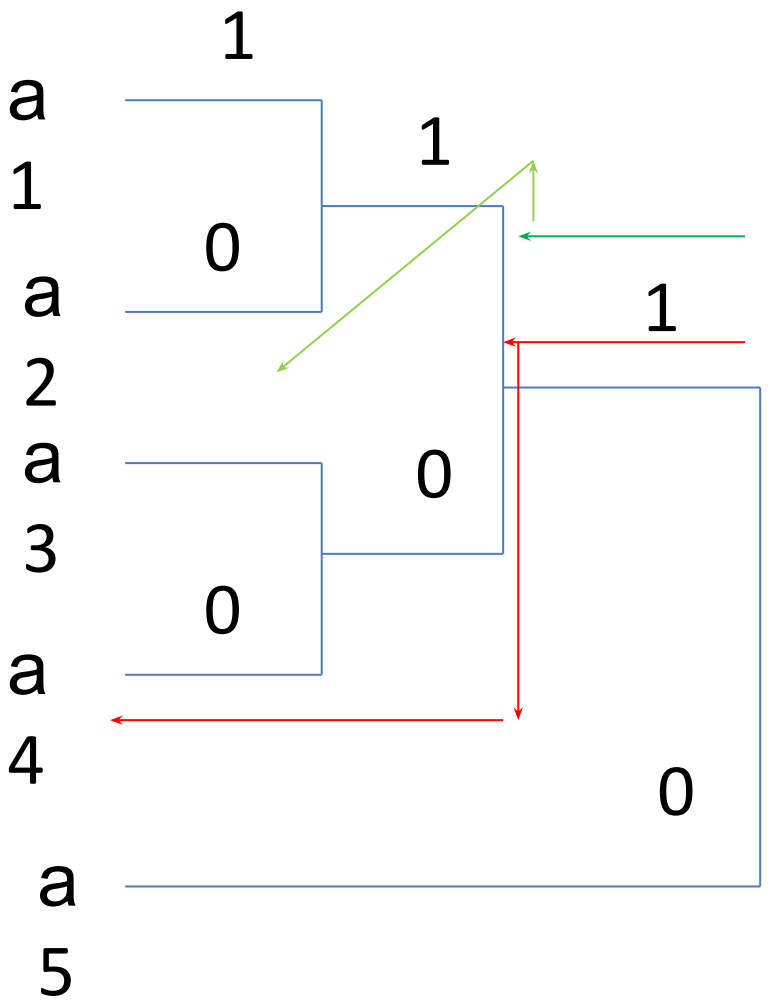
«признаки» оптимального дерева

- Объединение символов с минимальной вероятностью с символами с максимальной вероятностью

Когда не применим код Хаффмана

- Символы равновероятны
  - Если размер алфавита n является степенью 2, то получаются просто коды фиксированной длины. В других случаях коды весьма близки к кодам с фиксированной длиной
- Двухсимвольный алфавит
  - Идет потеря информации о корреляции соседних битов исходного изображения

Декодирование Хаффмана



Код 1001100111

100 110 0111

a4a2a5a

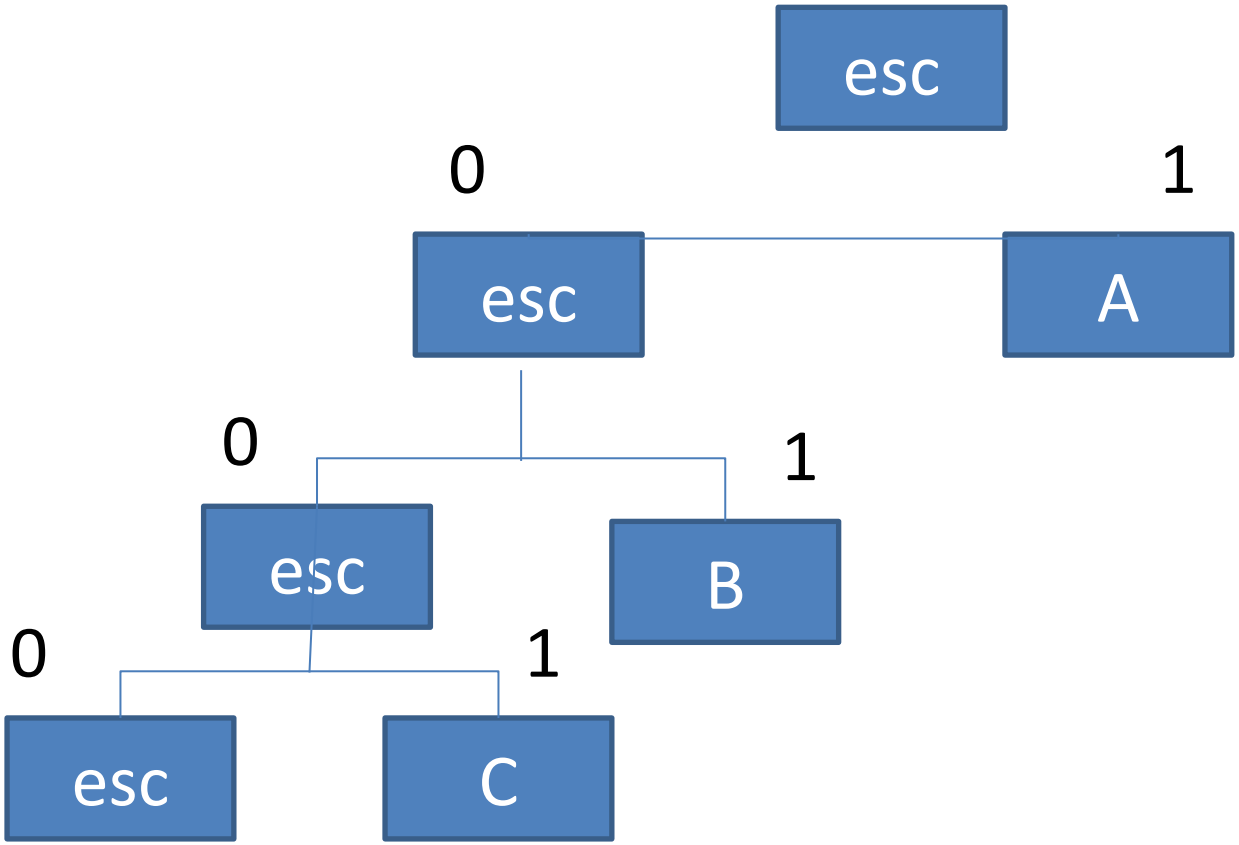
1

Адаптивное кодирование Хаффмана

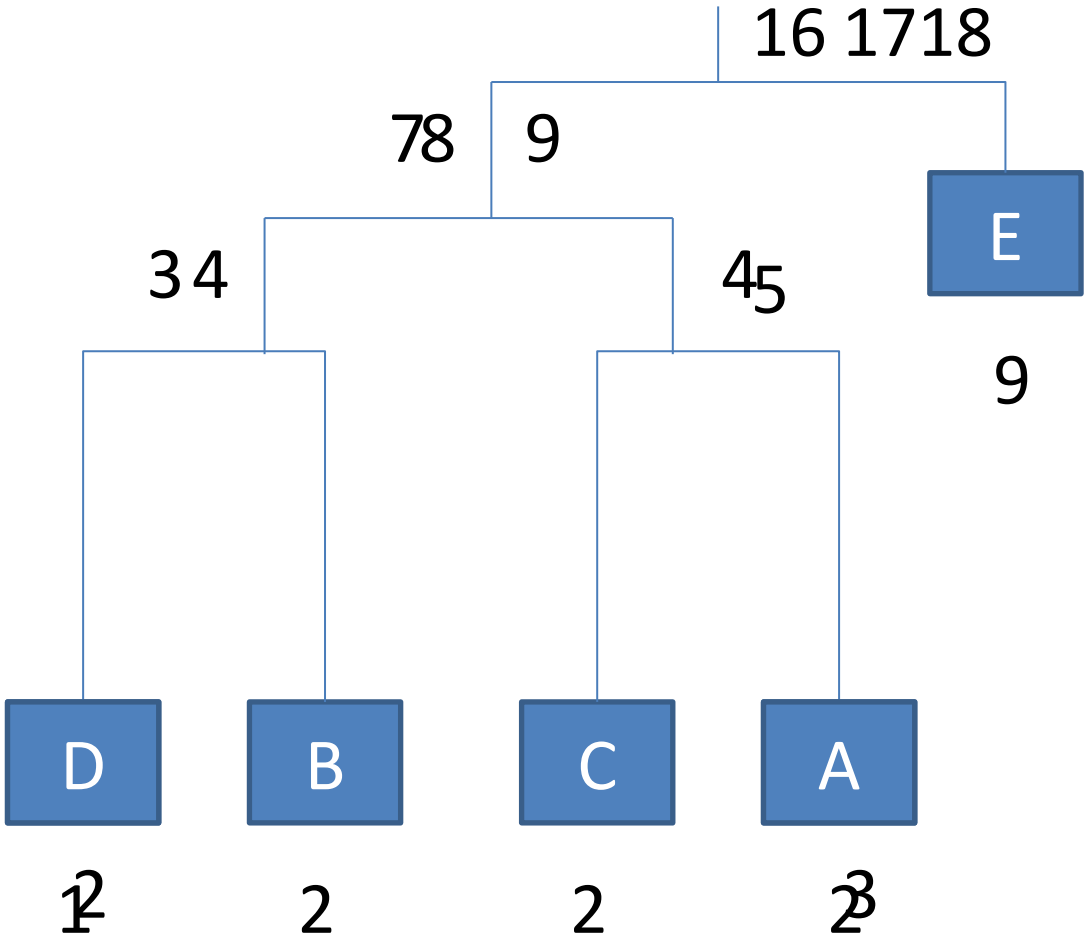
- До начала работы в дереве есть корень, в нем символ esc
- Новый символ просто добавляем в дерево без кодирования
- При повторе символа модифицируем дерево по принципу – частоты символов растут снизу вверх и слева направо

Добавляем новый символ

- Закодируем строку ABC



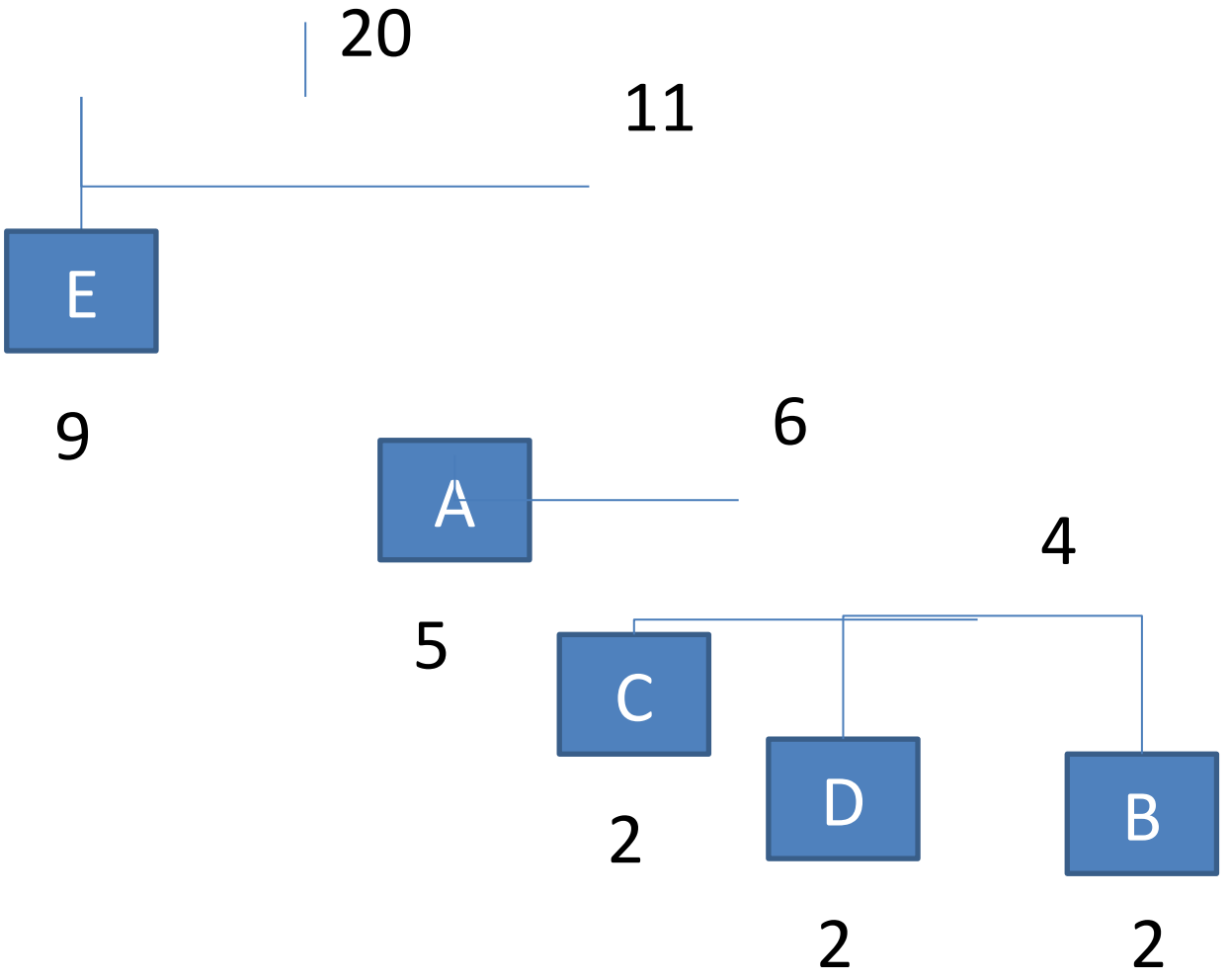
Модификация дерева



1. A++

2. A++

Модификация дерева



Арифметическое сжатие

- Всему кодируемому объекту назначается интервал [0;1)
- Вычисляются частоты появления символов входного алфавита в файле
- Начальный интервал делится пропорционально частотам символов
- По мере считывания символов из входного файла интервал переопределяется (сокращается)

пример

- Закодируем строку SWISS_MISS. Длина строки – 10 символов.
- Введем 2 переменные Low = 0, High = 1
- Частоты и интервалы символов

Символ	Частота	Интервал	Интервал	Накопленные частоты
S	5	5/10=0.5	[0.5;1)	5
W	1	1/10=0.1	[0.4;0.5)	4
I	2	2/10=0.2	[0.2;0.4)	2
M	1	1/10=0.1	[0.1;0.2)	1
_	1	1/10=0.1	[0.0;0.1)	0

Процесс кодирования

Символ	Low NewLow:=0ldLow+Range*LowRange(X)	High NewHigh:=0ldLow+Range*HighRange(X);
S	0.0+(1.0-0.0)x0.5 = 0.5	0.0+(1.0-0.0)x1.0 = 1.0
W	0.5+(1.0-0.5)x0.4 = 0.70	1.0+(1.0-0.5)x0.5 = 0.75
I	0.70+(0.75-0.70)x0.2 = 0.71	0.75+(0.75-0.70)x0.4 = 0.72
S	0.71+(0.72-0.71)x0.5 = 0.715	0.72+(0.72-0.71)x1.0 = 0.720
S	0.715+(0.720-0.715)x0.5 = 0.7175	0.720+(0.720-0.715)x1.0 = 0.7200
_	0.7175+(0.720-0.7175)x0.0 = 0.7175	0.7200+(0.720-0.7175)x0.1 = 0.71775
M	0.7175+(0.71775-0.7175)x0.1 = 0.717525	0.71775+(0.71775-0.7175)x0.2=0.717550
I	0.717525+(0.717550-0.7125)x0.2 = 0.717530	0.717550+(0.717550-0.717525)x0.4=0.717535
S	0.717530+(0.717535-0.717530)x0.5 = 0.7175325	0.717535+(0.717535-0.717530)x1=0.717535
S	0.717525+(0.717535-0.717525)x0.5 = 0.71753375 – конечный код	0.717535+(0.717535-0.717525)x1=0.717535

Декодирование

- Декодер узнает символы алфавита
- Получает информацию о частотах и интервалах символов
- Читает по 1 цифре из конечного кода

Процесс декодирования

Код	Интервал	Символ	Пересчет кода Code : = (Code-LowRange(X))/Range
0.7…	[0.5;1.0)	S	(0.71753375 - 0.5)/0.5=0.4350675
0.43…	[0.4;0.5)	W	(0.4350675 - 0.4)/0.1= 0.350675
0.35…	[0.2;0.4)	I	(0.350675-0.2)/0.2=0.753375
0.7…	[0.5;1.0)	S	(0.753375-0.5)/0.5=0.50675
0.5067..	[0.5;1.0)	S	(0.50675-0.5)/0.5=0.0135
0.01..	[0.0;0.1)	_	(0.0135-0.0)/0.1=0.135
0.13…	[0.1;0.2)	M	(0.135-0.1)/0.1=0.35
0.35	[0.2;0.4)	I	(0.35-0.2)/0.2=0.75
0.75	[0.5;1.0)	S	(0.75-0.5)/0.5=0.5
0.5	[0.5;1.0)	S	(0.5-0.5)/0.5=0.0

Несимметричное кодирование

- Если вероятности появления символов в строке очень разные, то есть опасность наступления 0 до конца строки при декодировании
- Для избежания этого добавляют специальный символ eof с очень низкой вероятностью

Особенности реализации

- Переменные Low и High делать целыми и хранить в них только часть после запятой
- При вычислении символа округлять до диапазона частот
- По мере накопления в левой части числа неменяющихся чисел, убирать их