Алгоритмы и структуры данных

Лекция 8

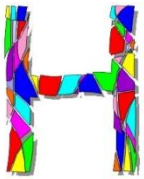
ДЕРЕВЬЯ ХАФФМАНА

(продолжение)

Динамическое кодирование по Хаффману

Фоллер (Newton **F**aller) [1973] и Галлагер (Robert **G**allager) [1978]. Кнут (Donald **K**nuth) [1985]

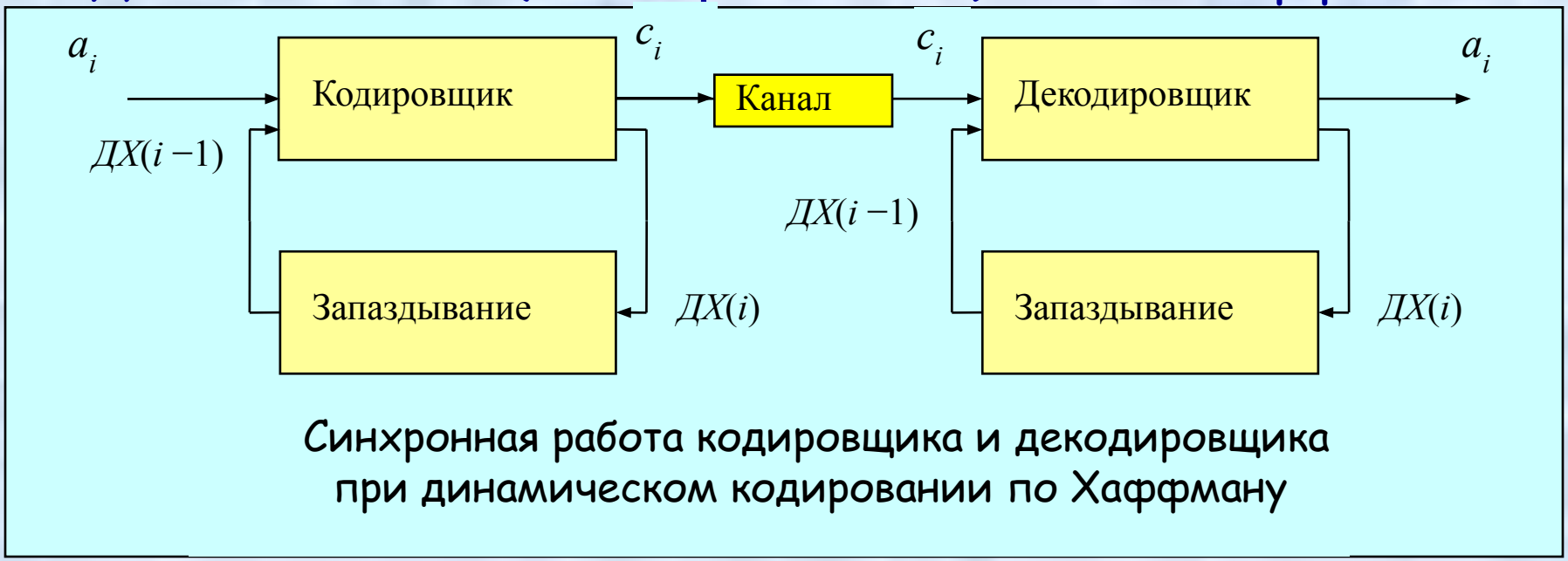
Динамическое кодирование по Хаффману



Недостатки *статического* метода Хаффмана :

- *двухпроходность* алгоритма (сначала вычисляются веса , затем строится дерево (код));
- необходимость передавать кодовое дерево вместе с последовательностью закодированных сообщений.

Динамический (однопроходный) метод Хаффмана



Синхронная работа кодировщика и декодировщика при динамическом кодировании по Хаффману

Кодировщик	Декодировщик
1) построение кода c_i для символа a_i по дереву Хаффмана $ДХ(i-1)$; 2) перестройку $ДХ(i-1)$ в новое $ДХ(i)$ с учетом поступившего символа a_i	1) восстановление символа a_i по коду c_i и по дереву $ДХ(i-1)$; 2) перестройку $ДХ(i-1)$ в новое $ДХ(i)$ с учетом декодированного символа a_i

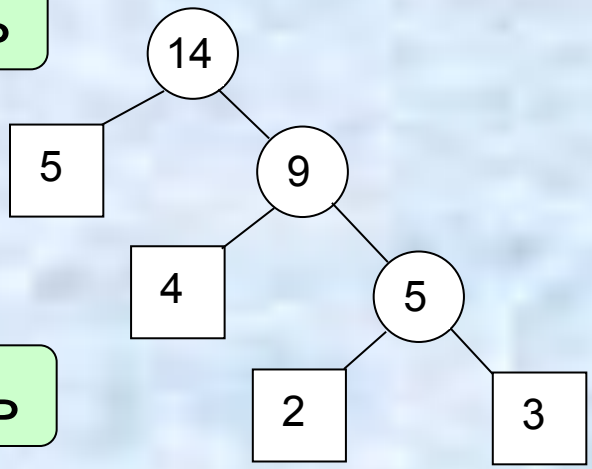
Алгоритм перестроения дерева Хаффмана

Пусть дерево Хаффмана (ДХ) строится так, что левый сын имеет вес не больший, чем правый.

При этом ДХ, вообще говоря, не единственно.

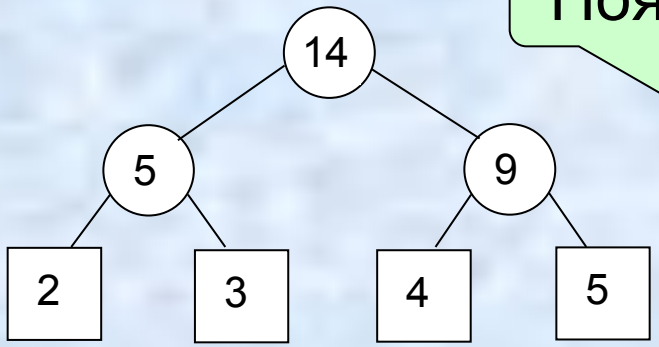
Пример: заданы веса $W_4 = (5, 4, 3, 2)$.

Пояснить 1



(2, 3, 4, 5, 5, 9, 14)

Пояснить 2



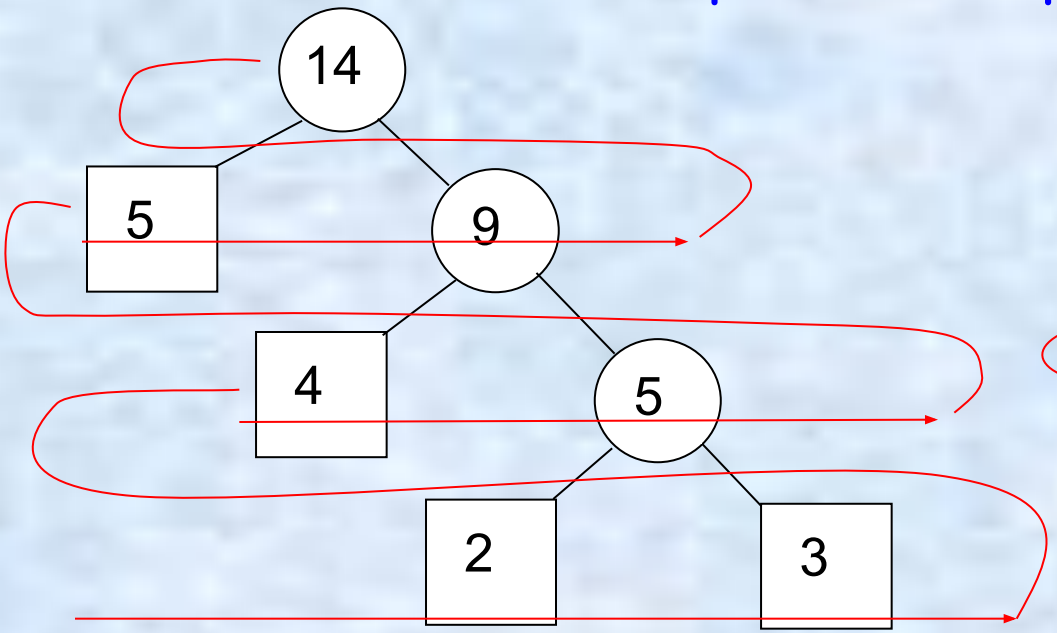
(2, 3, 4, 5, 5, 9, 14)

добавление a_1,
$w_1 \leftarrow w_1 + 1$ $(6 \leftarrow 5 + 1)$

добавление a_4,
$w_4 \leftarrow w_4 + 1$ $(3 \leftarrow 2 + 1)$

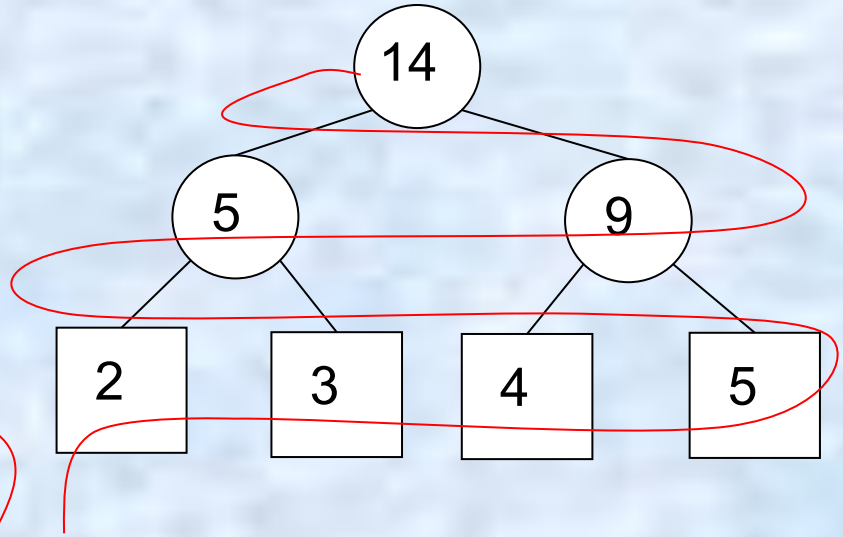
Пояснения:

левый вариант



(2, 3, 4, 5, 5, 9, 14)

правый вариант



(2, 3, 4, 5, 5, 9, 14)

Дерево Хаффмана является *строго бинарным* и содержит ровно $2n - 1$ узлов, n из которых являются листьями.

Действительно, *пусть* в строго бинарном дереве содержится n листьев (внешних узлов) и s внутренних узлов.

Тогда число *исходящих* из внутренних узлов веток есть $2s$.

Подсчет этих же веток, как *входящих* во все узлы дерева, кроме корня, дает $n + s - 1$.

Таким образом, $2s = n + s - 1$.

Отсюда следует $s = n - 1$

и общее число узлов $n + s = 2n - 1$.

В общем случае неубывающая последовательность весов $x_1 \le x_2 \le x_3 \le \ldots \le x_{2n-1}$, получаемых в порядке взвешенного сочетания узлов (т.е. порядке выбора минимальных весов и порождения суммарного веса) в алгоритме Хаффмана, инвариантна для всех деревьев Хаффмана с заданными весами $w_1 \ge w_2 \ge \ldots \ge w_{n-1} \ge w_n$.

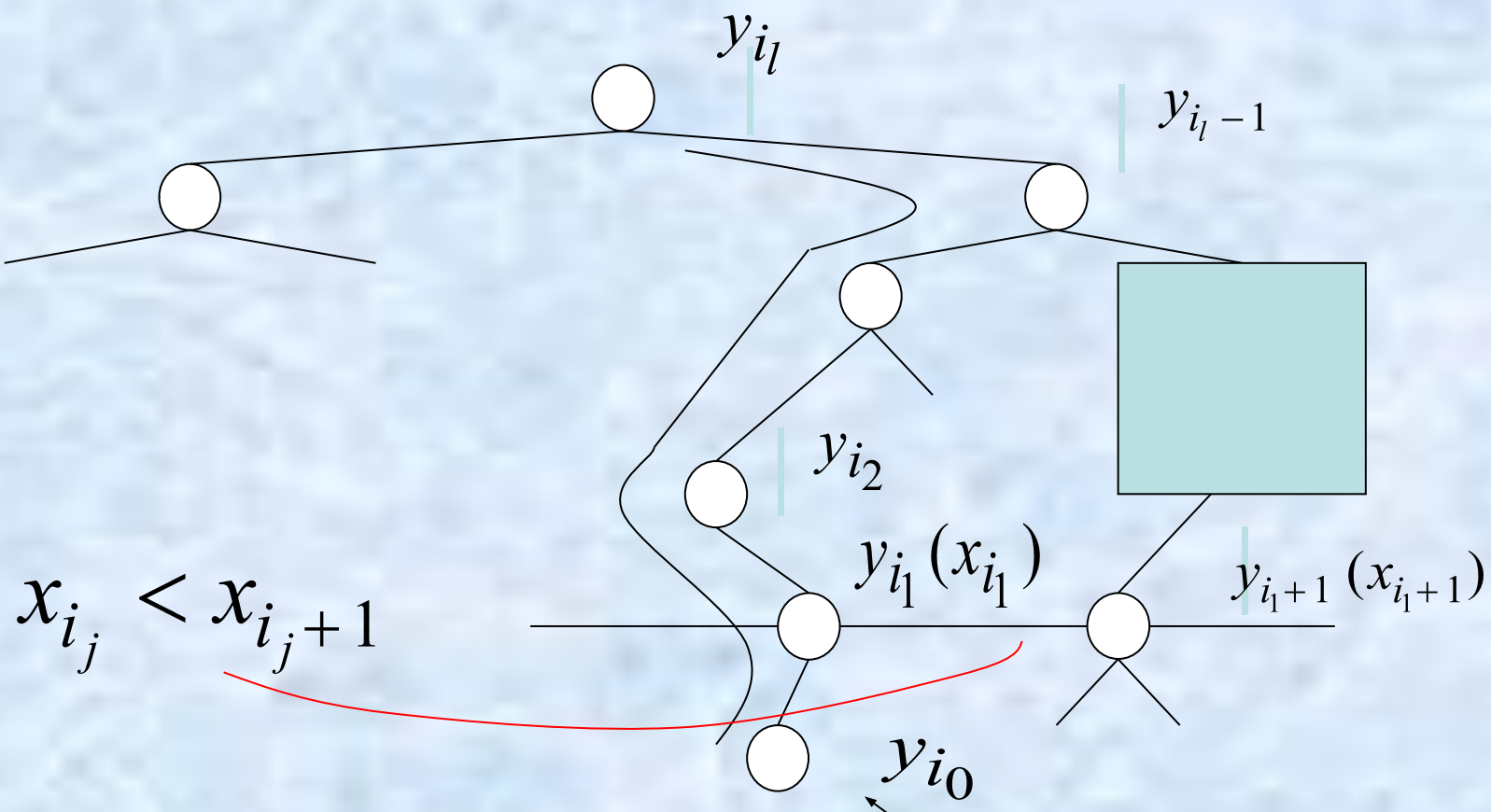
При этом внутренние узлы имеют веса $x_1 + x_2, x_3 + x_4, \ldots, x_{2n-3} + x_{2n-2} = x_{2n-1}$.

Строго бинарное дерево – упорядоченное, если:

А) n внешних узлов (листьев) получили веса $(w_1, w_2, \ldots, w_n)$ в каком-то порядке, и каждый внутренний узел получил вес, равный сумме своих сыновей;

Б) $2n - 1$ узлов (внутренних и внешних) можно перечислить в такой последовательности $(y_1, y_2, \ldots, y_{2n-1})$, что если x_i – *вес узла* y_i, то $x_1 \le x_2 \le \ldots \le x_{2n-1}$; узлы y_{2j-1} и y_{2j} – братья (сыновья одного отца).

упорядоченное дерево ⇔ дерево Хаффмана



Из различных деревьев Хаффмана можно выбрать такое, которое не изменится при модификации веса $w \leftarrow w + 1$

Пример обеспечения свойства модифицируемости

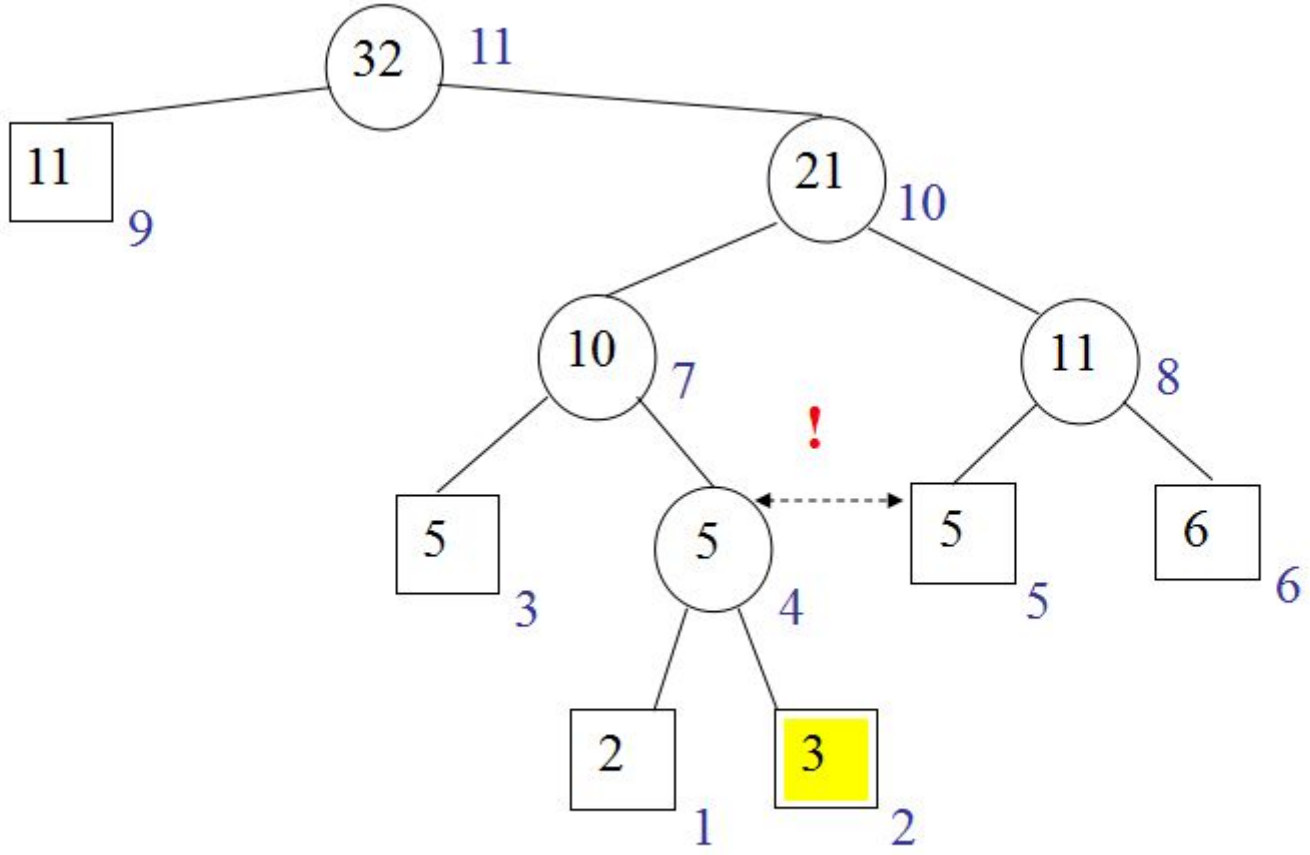
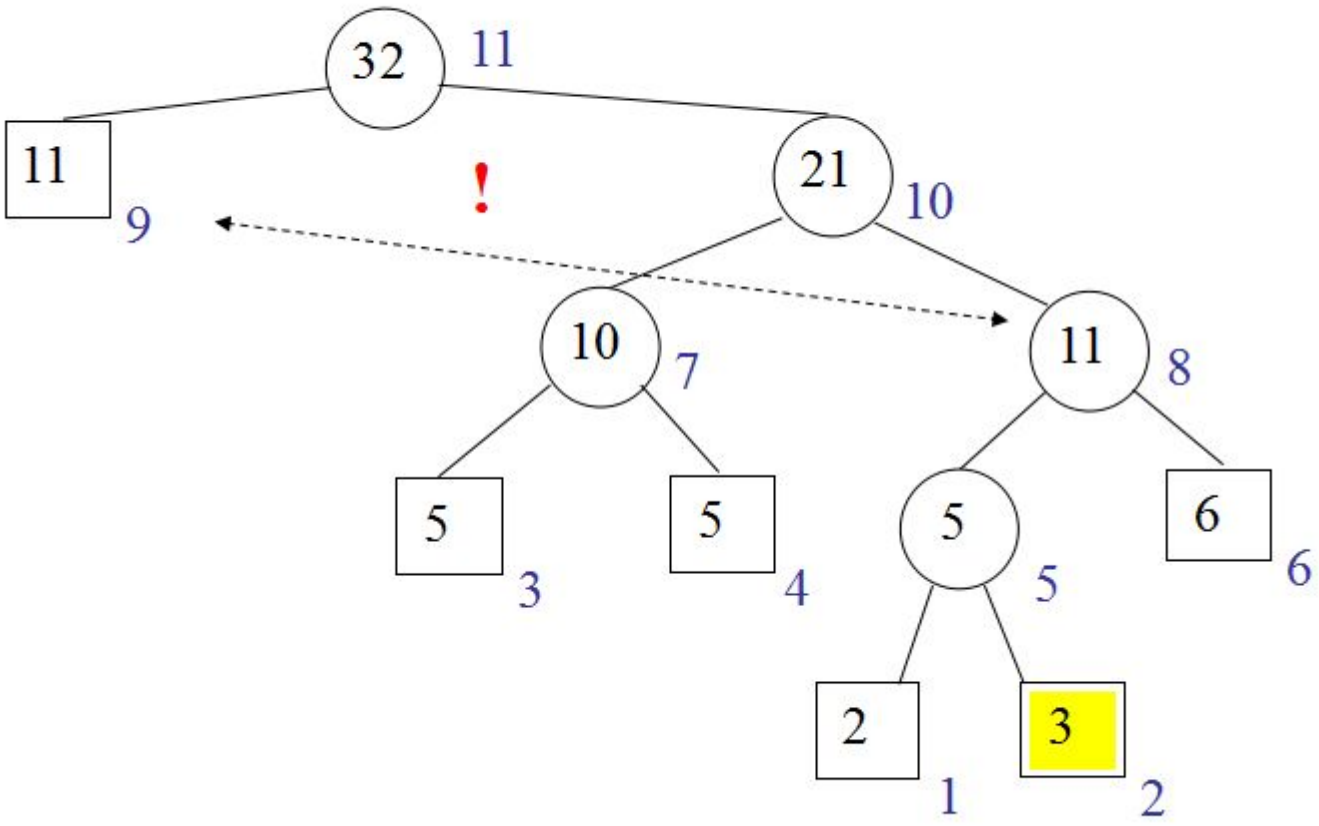
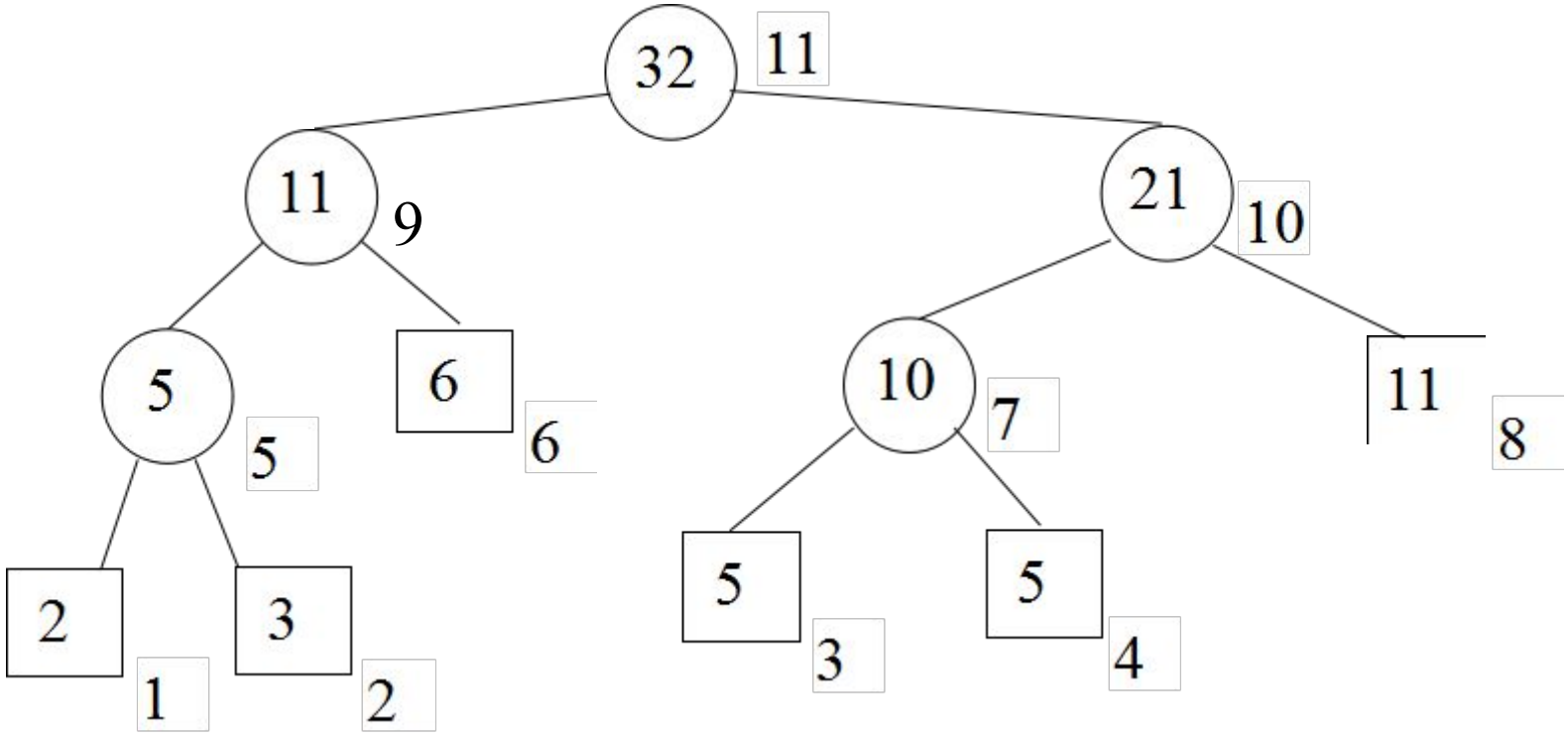


Рис. Пример дерева Хаффмана

Инвариант: (2, 3, 5, 5, 5, 6, 10, 11, 11, 21, 32),

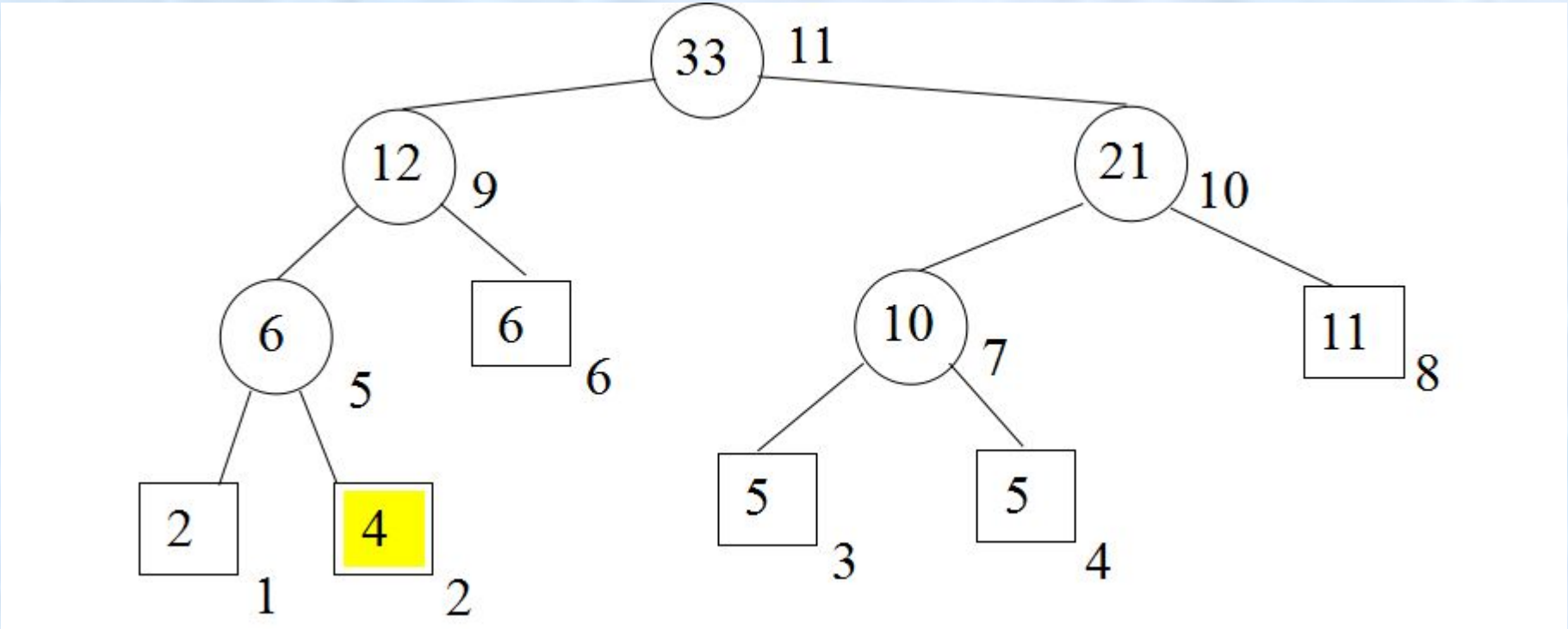

32
11
11
9
!
21
10
10
7
11
8
5
3
5
4
5
5
6
6
2
1
3
2



Это дерево также имеет инвариант
(2, 3, 5, 5, 5, 6, 10, 11, 11, 21, 32), и теперь *на всем пути*

⟨2, 5, 9, 11⟩ *к корню* требуемые неравенства выполнены

Окончательно имеем



В итоговом алгоритме перевешивания и коррекции весов происходят в одном цикле на пути от листа к корню

АБРАКАДАБРА!

Пример динамического кодирования

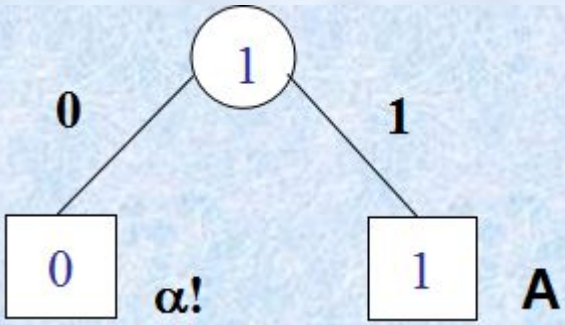
Обозначим α! – узел нулевого веса, отображающий любой символ алфавита, не встречающиеся до сих пор.

Первые вхождения символов кодируются кодом узла α! , за которым следует код нового символа в специальной кодировке. Пока обозначим такой код подчёркиванием.

Перед началом кодирования и кодировщик и декодировщик знают лишь алфавит. Важно количество символов алфавита для кодирования первых вхождений.

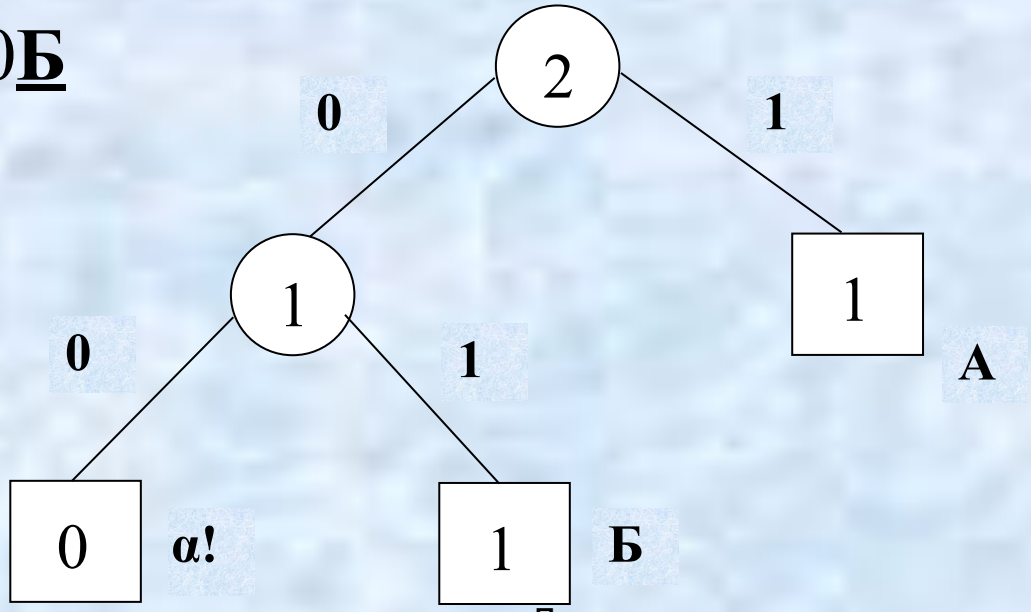
$А_1Б_2Р_3А_4К_5А_6Д_7А_8Б_9Р_{10}А_{11}!_{12}$

$А \Rightarrow \underline{А}$



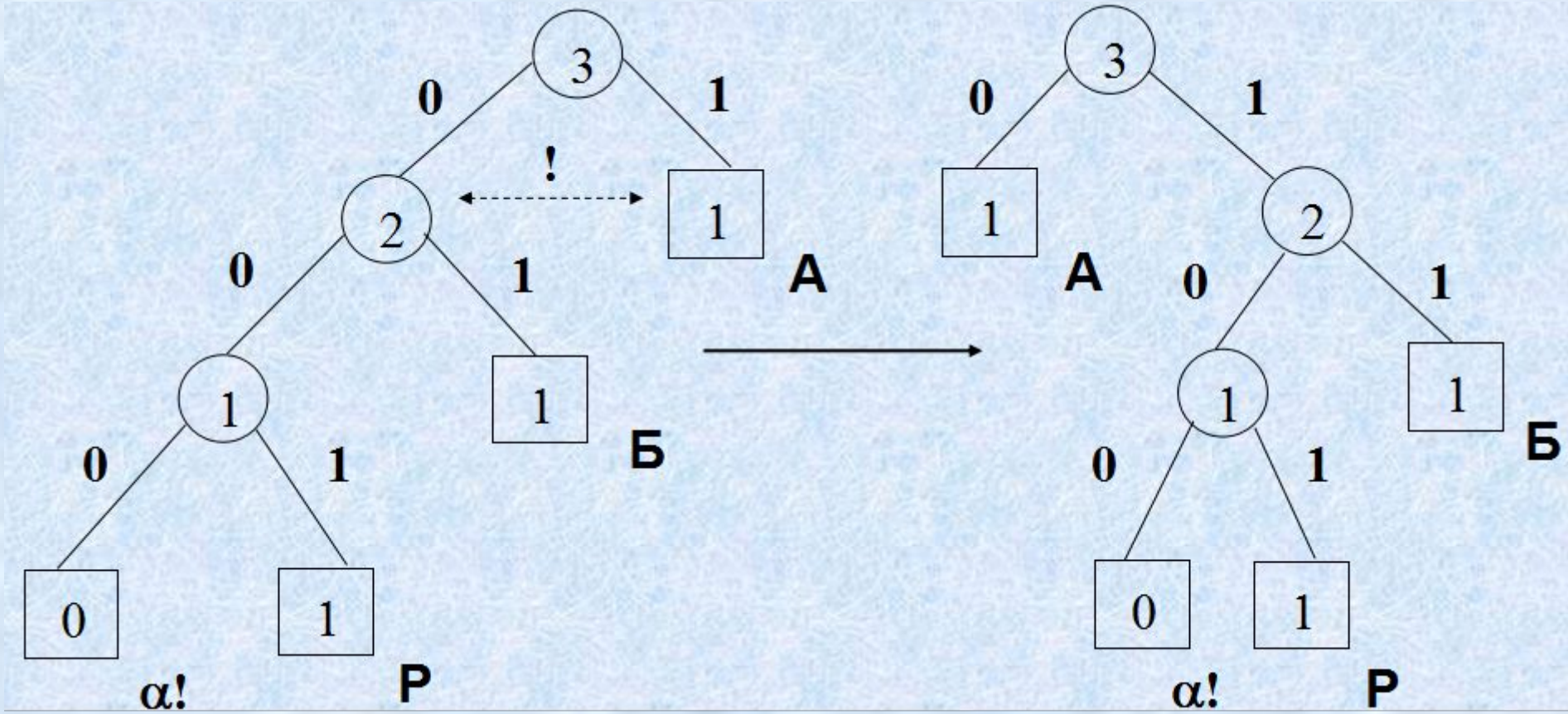
$А_1Б_2Р_3А_4К_5А_6Д_7А_8Б_9Р_{10}А_{11}!_{12}$

$Б \Rightarrow 0\underline{Б}$



$А_1Б_2Р_3А_4К_5А_6Д_7А_8Б_9Р_{10}А_{11}!_{12}$

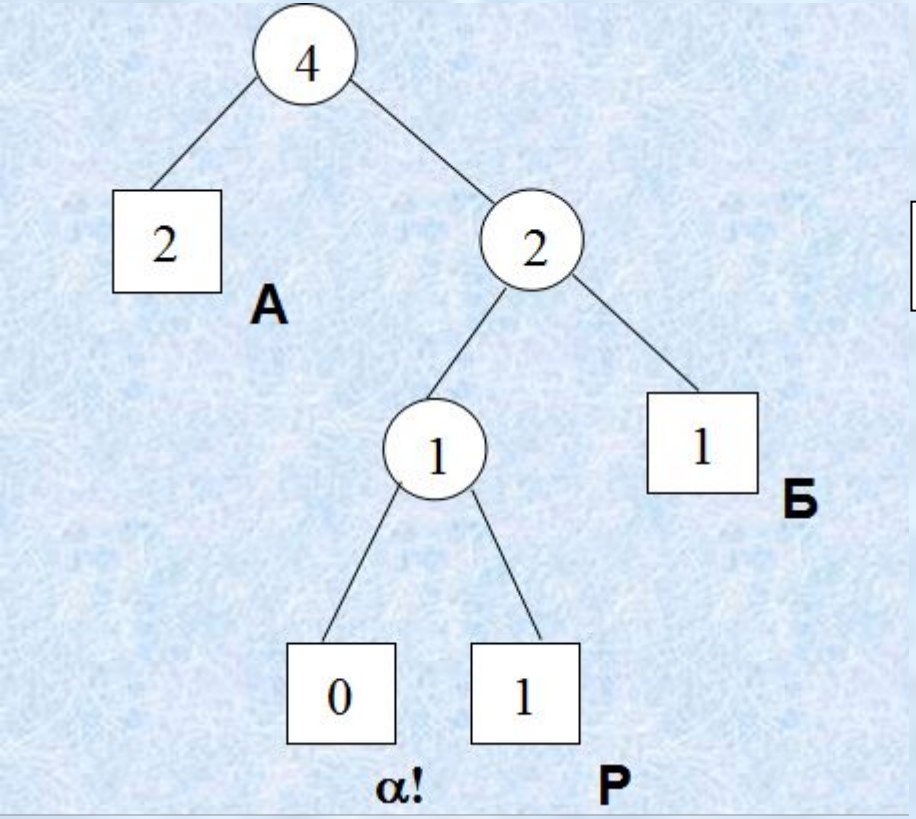
Р ⇒ 00**Р**

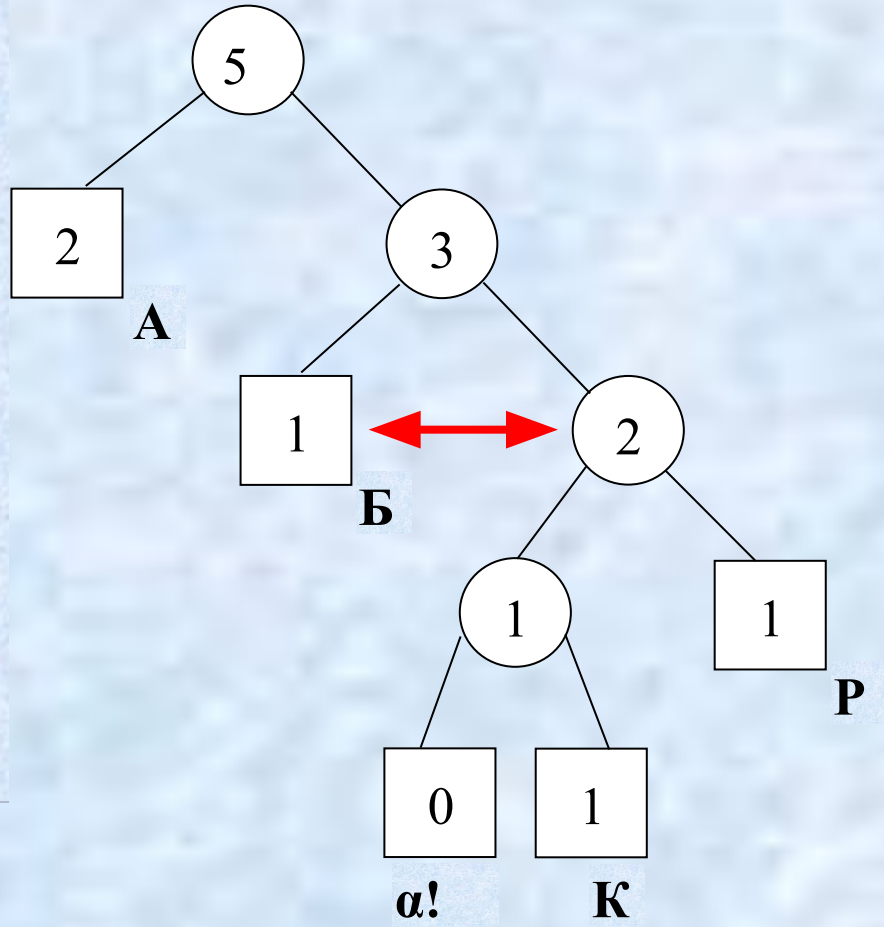


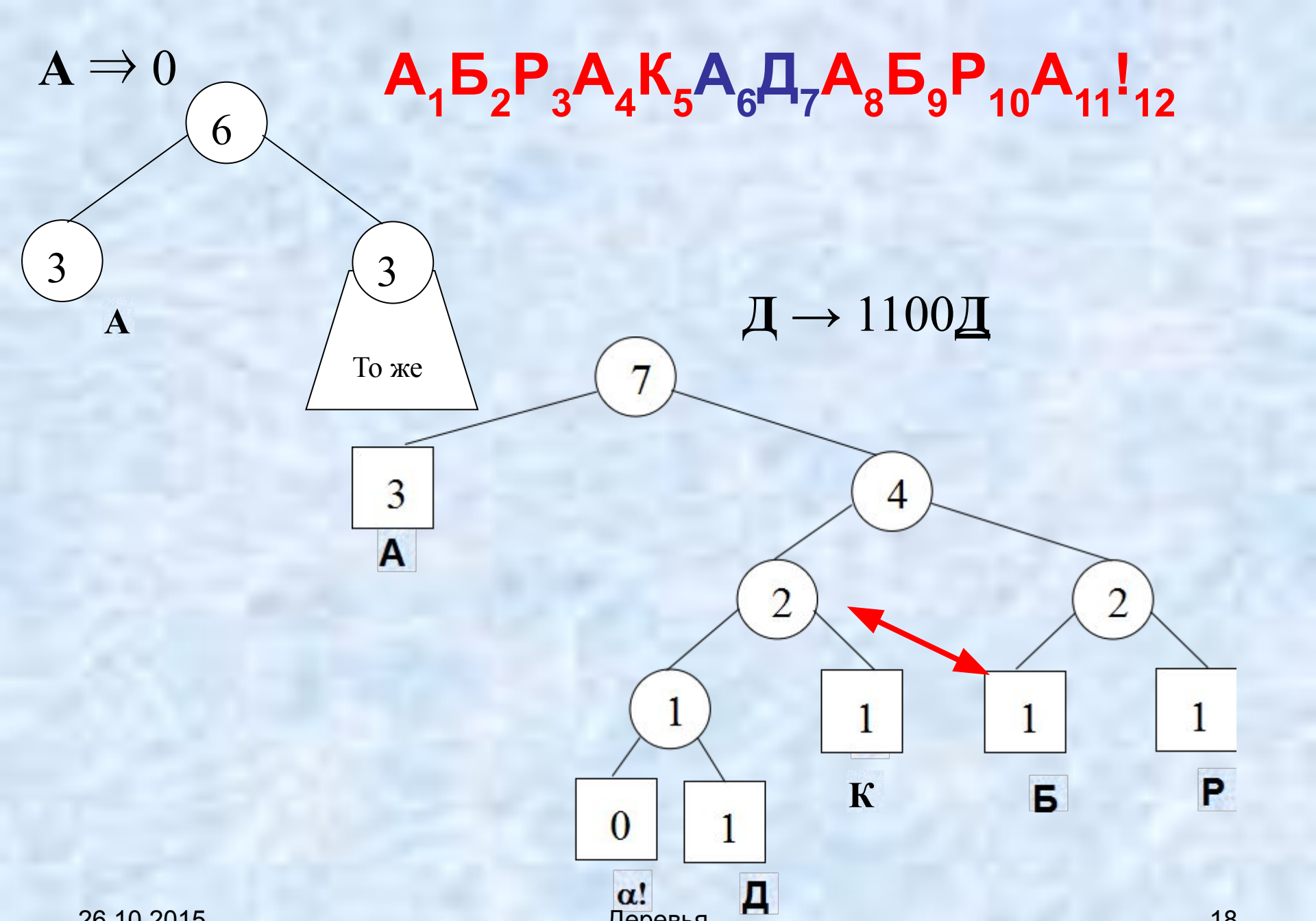
$А_1Б_2Р_3А_4К_5А_6Д_7А_8Б_9Р_{10}А_{11}!_{12}$

$\mathbf{А} \Rightarrow 0$

$\mathbf{К} \Rightarrow 100\ \underline{\mathbf{К}}$



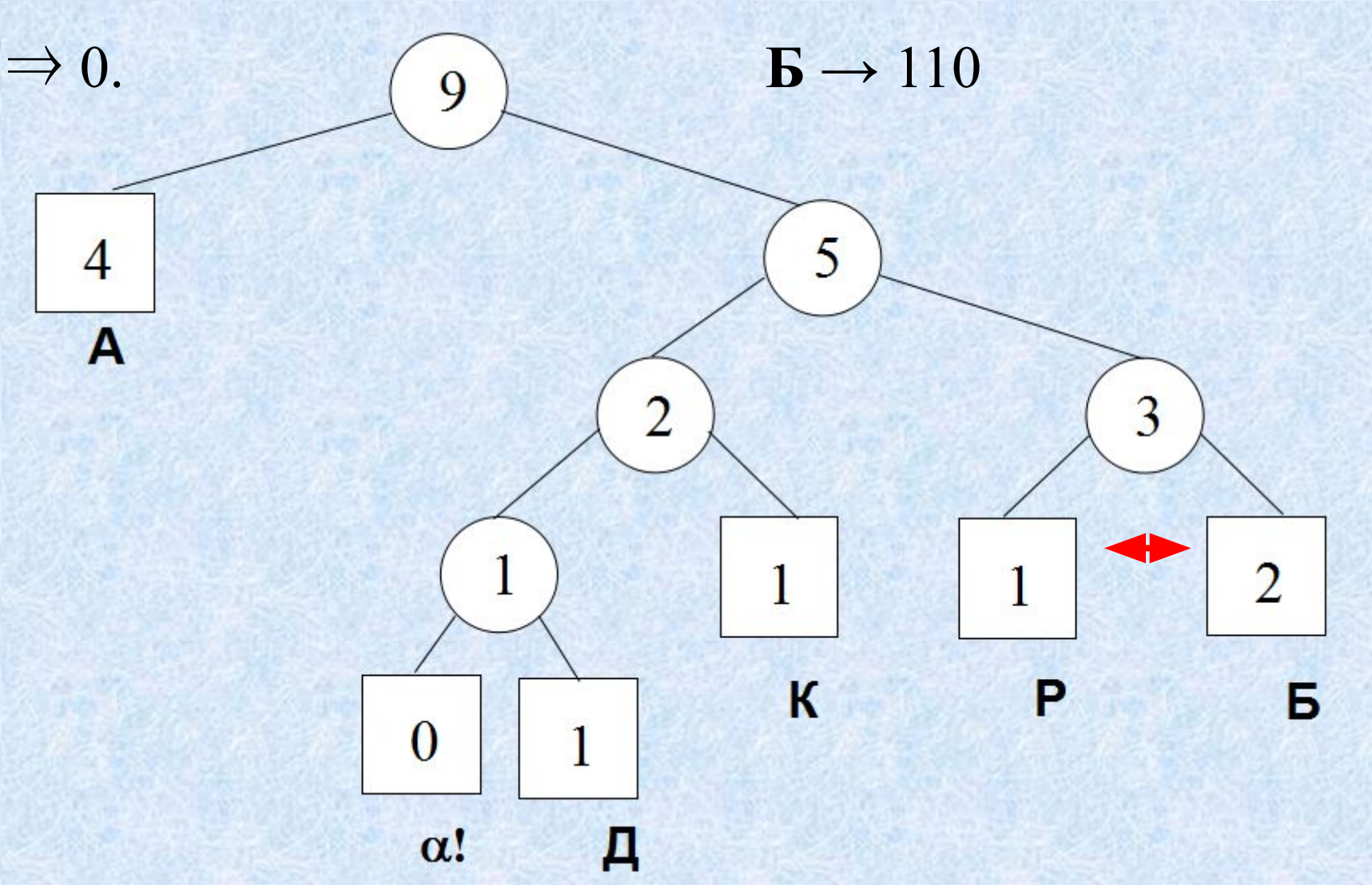



А ⇒ 0
А1Б2Р3А4К5А6Д7А8Б9Р10А11!12
6
3
3
А
То же
Д → 1100Д
7
3
А
4
2
2
1
1
1
1
0
1
α!
Д
К
Б
Р

$А_1Б_2Р_3А_4К_5А_6Д_7А_8Б_9Р_{10}А_{11}!_{12}$

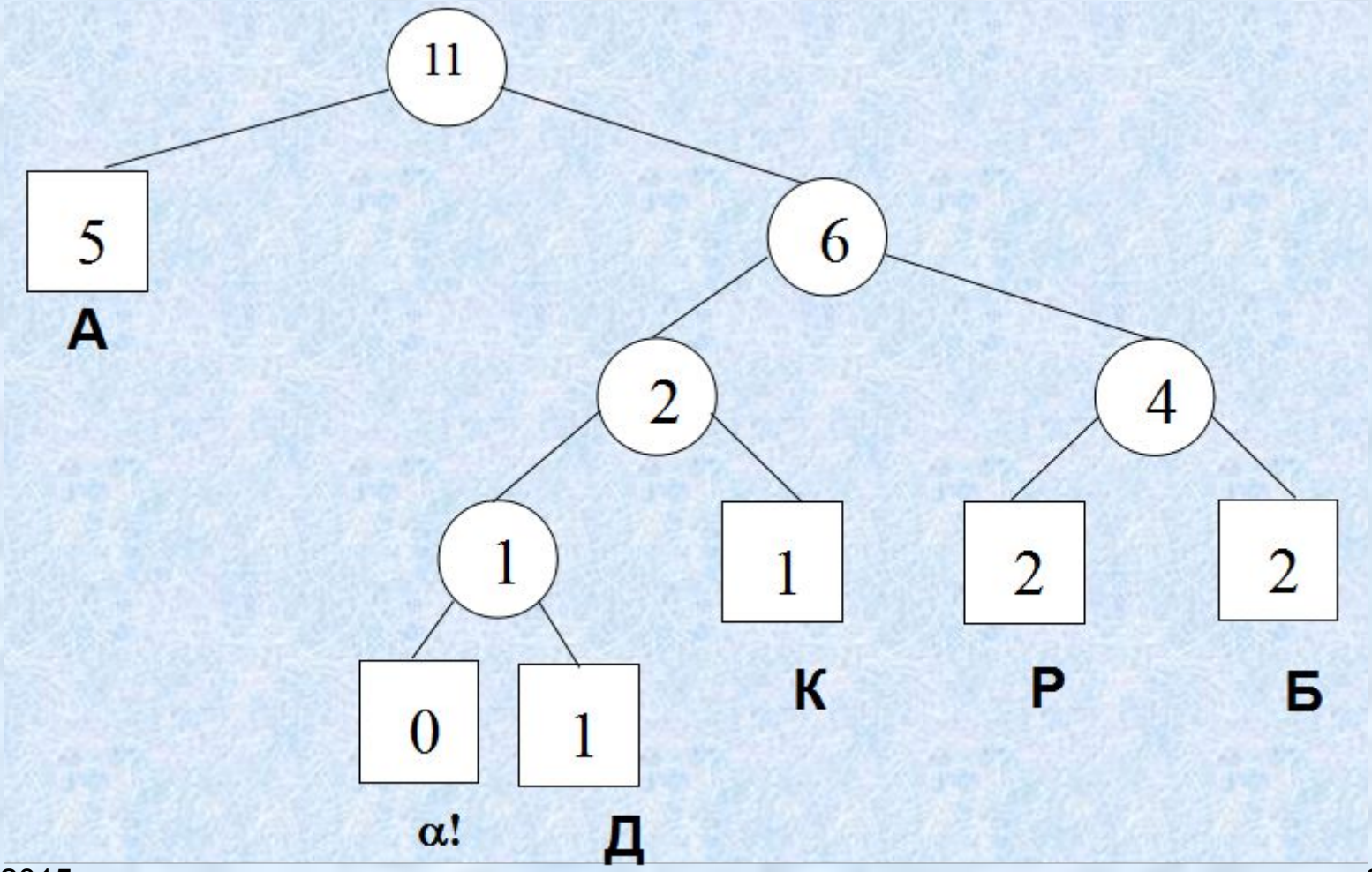
А $\Rightarrow 0.$

Б $\rightarrow 110$



$А_1Б_2Р_3А_4К_5А_6Д_7А_8Б_9Р_{10}А_{11}!_{12}$

Р ⇒ 110 (ср. шаг 9 !) **А** ⇒ 0



$А_1Б_2Р_3А_4К_5А_6Д_7А_8Б_9Р_{10}А_{11}!_{12}$

! ⇒1000!. Конец текста.

Перестраивать дерево не требуется.

А 0**Б** 00**Р** 0 100**К** 0 1100**Д** 0 110 110 0 1000!

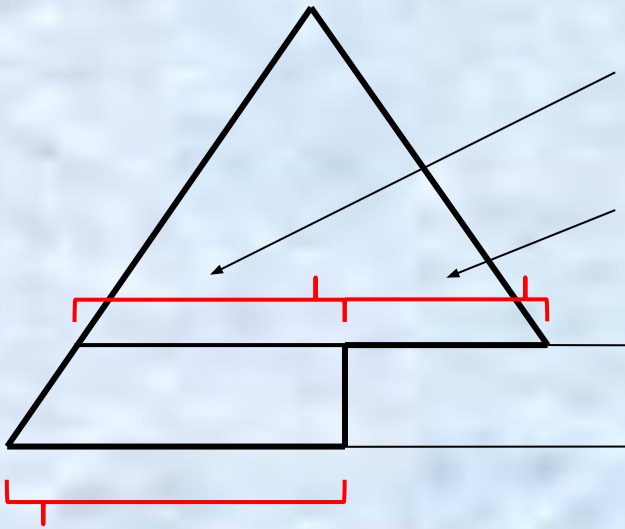
↑ ↑ ↑ ↑ ↑ ↑

А А А Б Р А

Кодирование первых вхождений

$$n = 2^p + q$$

$$0 \le q < 2^p$$



q внутренних узлов (отцов $2q$ листьев)

$2^p - q = n - 2q$ листьев

p-й уровень

$(p + 1)$-й уровень

$2q$ листьев

Если $1 \le i \le 2q$, то символ a_i кодируется как $(p + 1)$-битное представление числа $i - 1$,

иначе – как p-битное представление числа $i - q - 1$.

Входной алфавит: **АБВГДЕЁЖ…ЭЮЯ!**

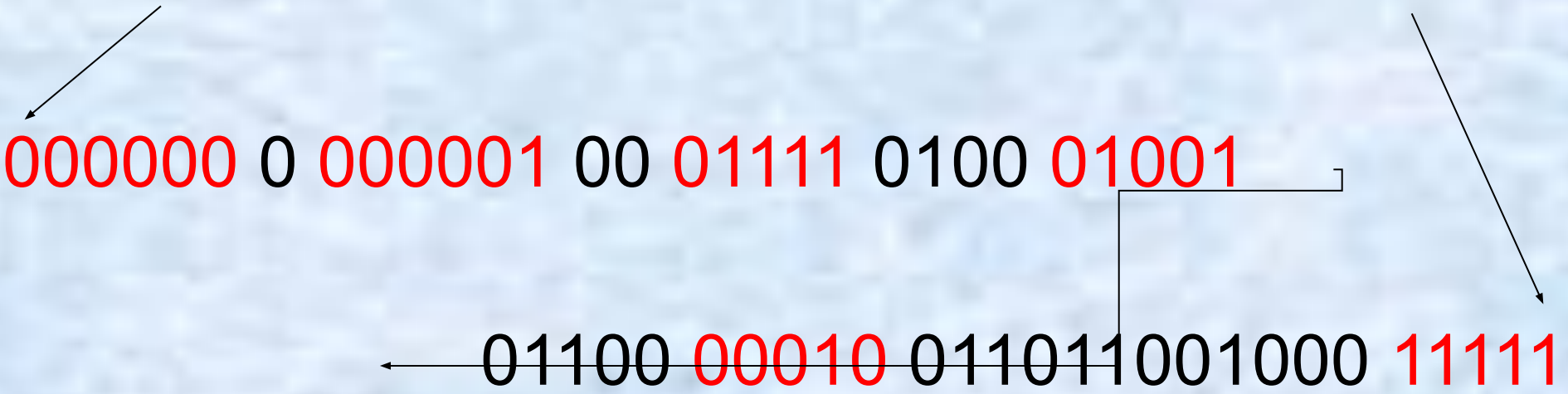
***n* = 34** $\rightarrow$ $34 = 2^5 + 2$, т. е. $p = 5$ и $q = 2$

- **А**, **Б**, **В** и **Г** кодируются 6-битными кодами 000000, 000001, 000010 и 000011
- остальные 30 букв кодируются 5-битными кодами чисел от 2 до 31, т. е. кодами от 00010 до 11111

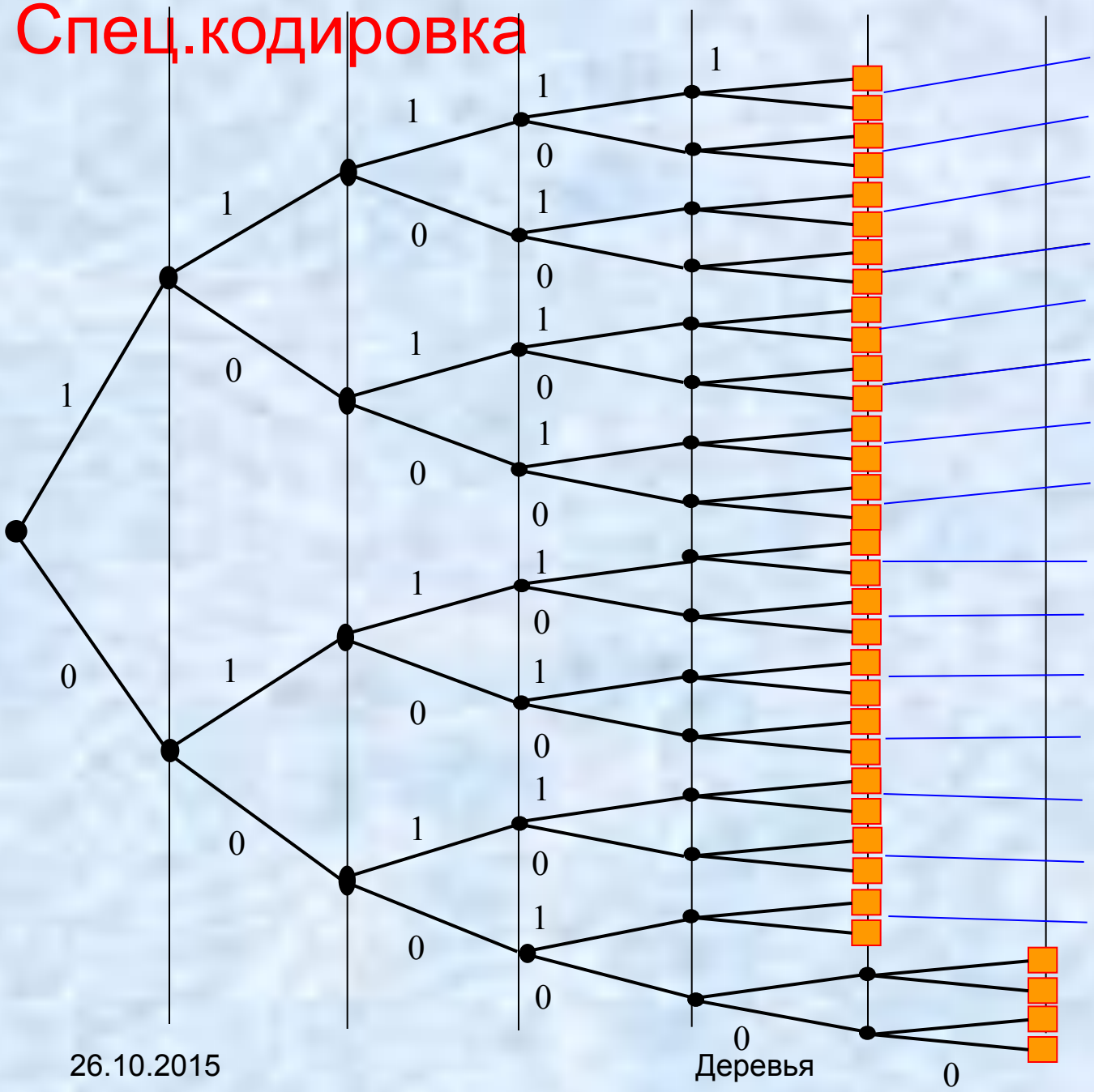
код (**А**) = 000000 *код* (**Б**) = 000001 *код* (**Д**) = 00010	*код* (**К**) = 01001 *код* (**Р**) = 01111 *код* (**!**) = 11111

А0**Б**00**Р**0100**К**01100**Д**011011001000**!**

А0**Б**00**Р**0100**К** 01100**Д**011011001000**!**



Спец.кодировка



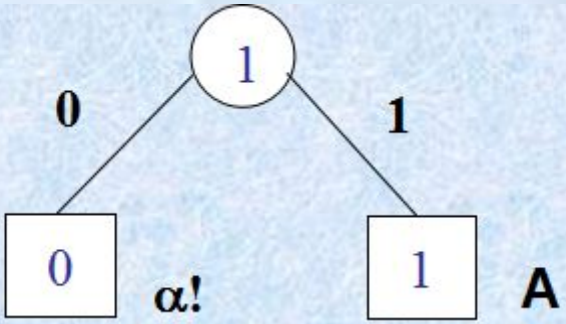
!	1 1 1 1 1
Я	1 1 1 1 0
Ю	1 1 1 0 1
Э	1 1 1 0 0
Ь	1 1 0 1 1
Ы	1 1 0 1 0
Ъ	1 1 0 0 1
Щ	1 1 0 0 0
Ш	1 0 1 1 1
Ч	1 0 1 1 0
Ц	1 0 1 0 1
Х	1 0 1 0 0
Ф	1 0 0 1 1
У	1 0 0 1 0
Т	1 0 0 0 1
С	1 0 0 0 0
Р	0 1 1 1 1
П	0 1 1 1 0
О	0 1 1 0 1
Н	0 1 1 0 0
М	0 1 0 1 1
Л	0 1 0 1 0
К	0 1 0 0 1
Й	0 1 0 0 0
И	0 0 1 1 1
З	0 0 1 1 0
Ж	0 0 1 0 1
Ё	0 0 1 0 0
Е	0 0 0 1 1
Д	0 0 0 1 0
Г	0 0 0 0 1 1
В	0 0 0 0 1 0
Б	0 0 0 0 0 1
А	0 0 0 0 0 0

Декодирование

000000000000100011110100010010110000010011011001000011111

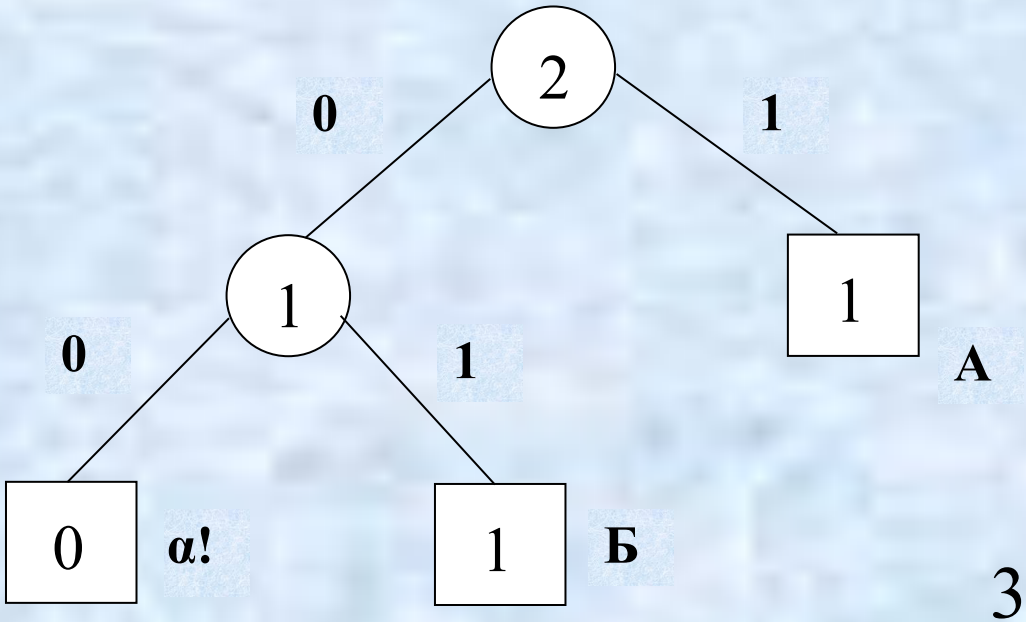
1) В спец. кодировке: 000000 - A

000000000000100011110100010010110000010011011001000011111



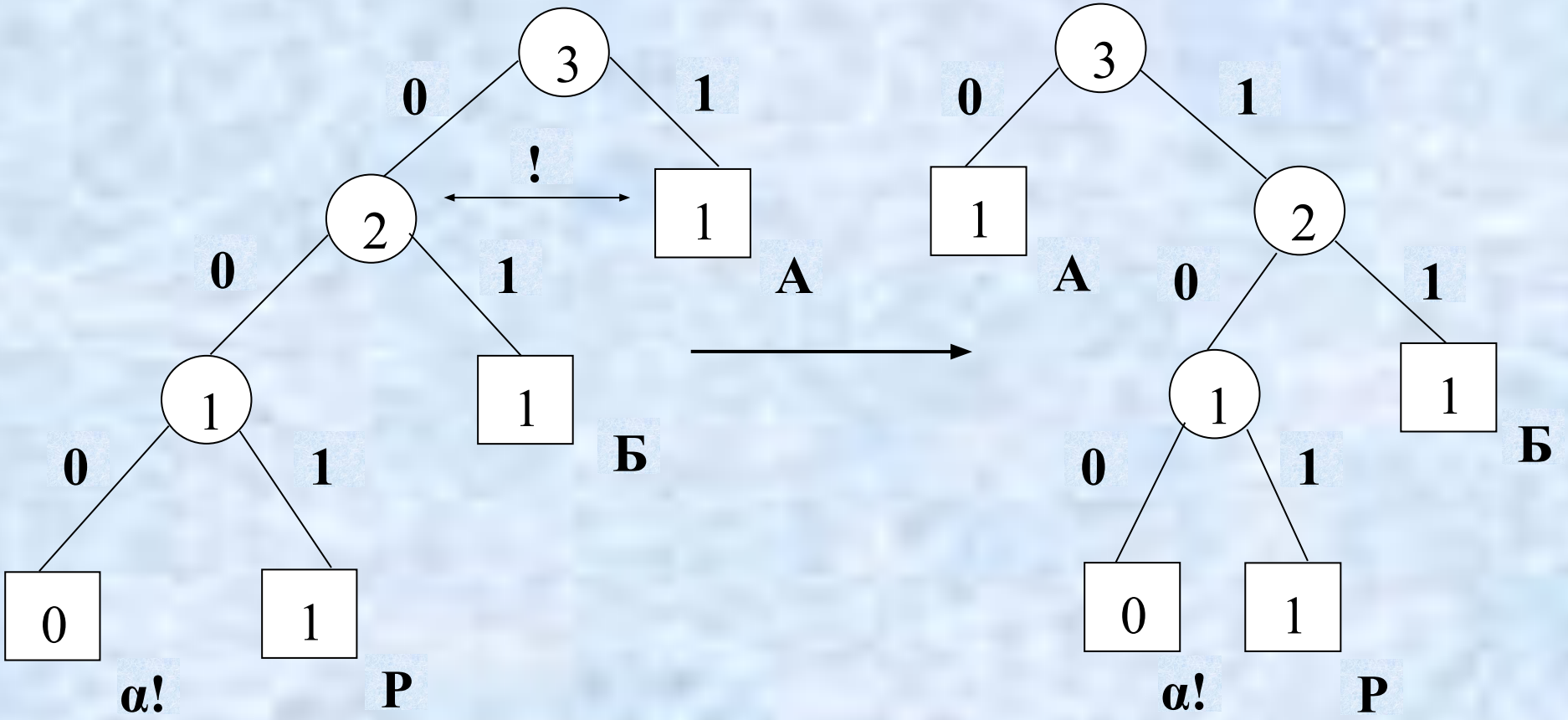
2) α! + Б

000000**0***000001*0001111010001001011000001001101100100011111



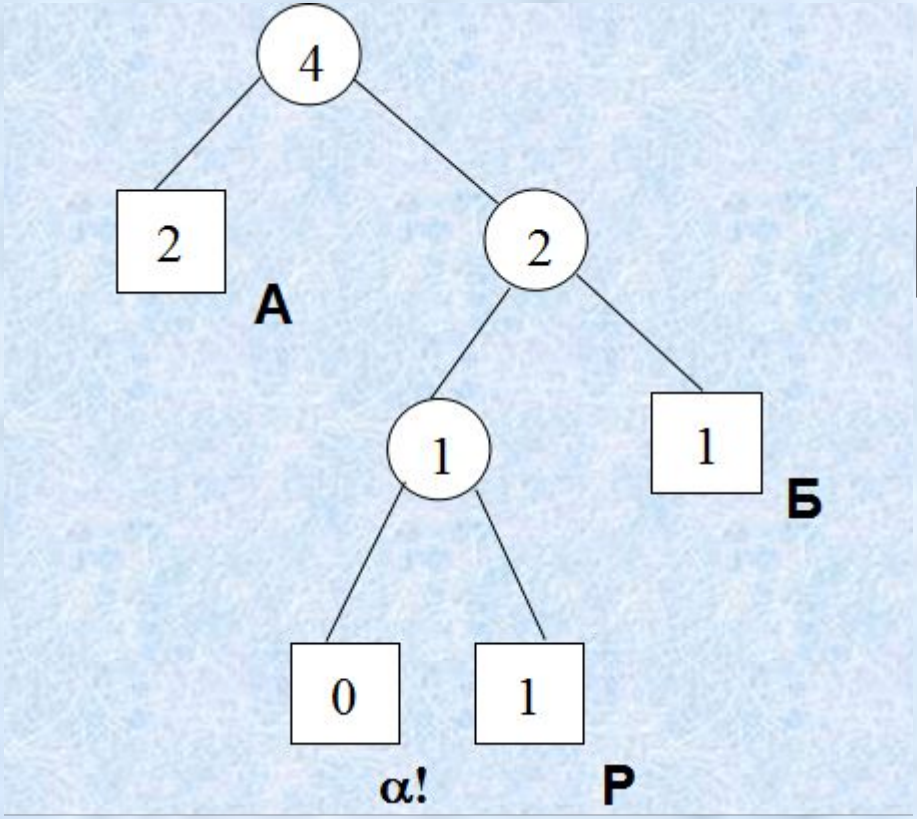
3) α! + Р → АБР

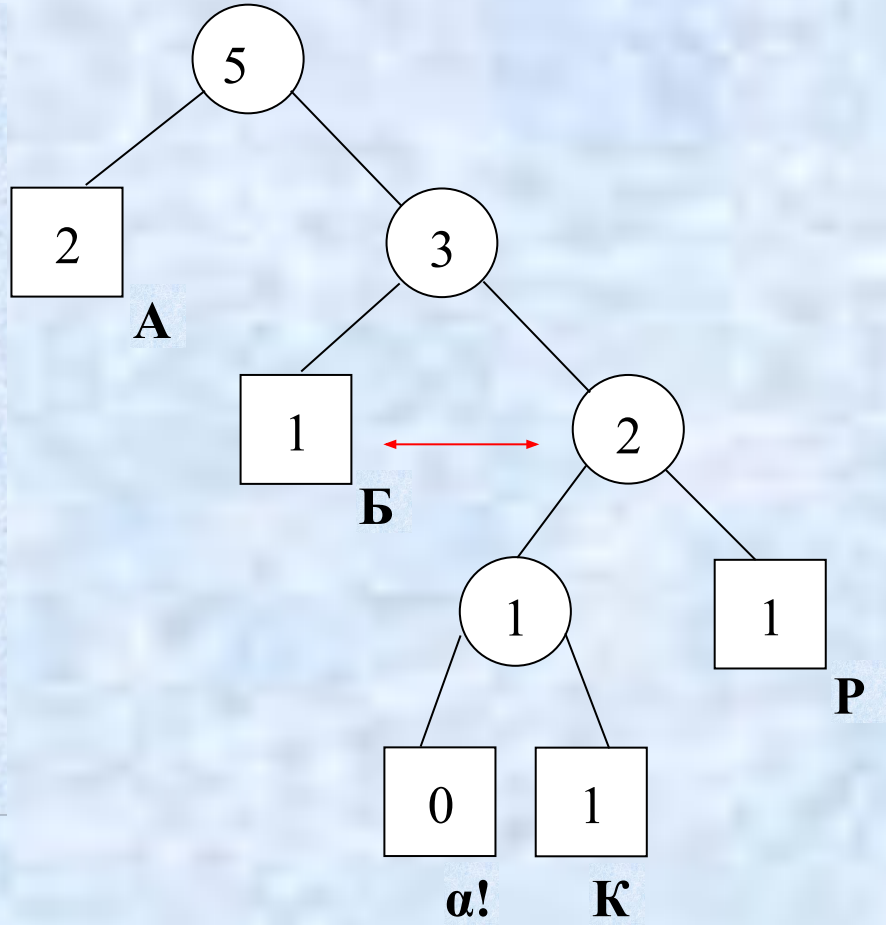
000000000000**00***01111*0100010010110000010011011001000111111



4) А (вес(А) 1→2) 5) α! + К

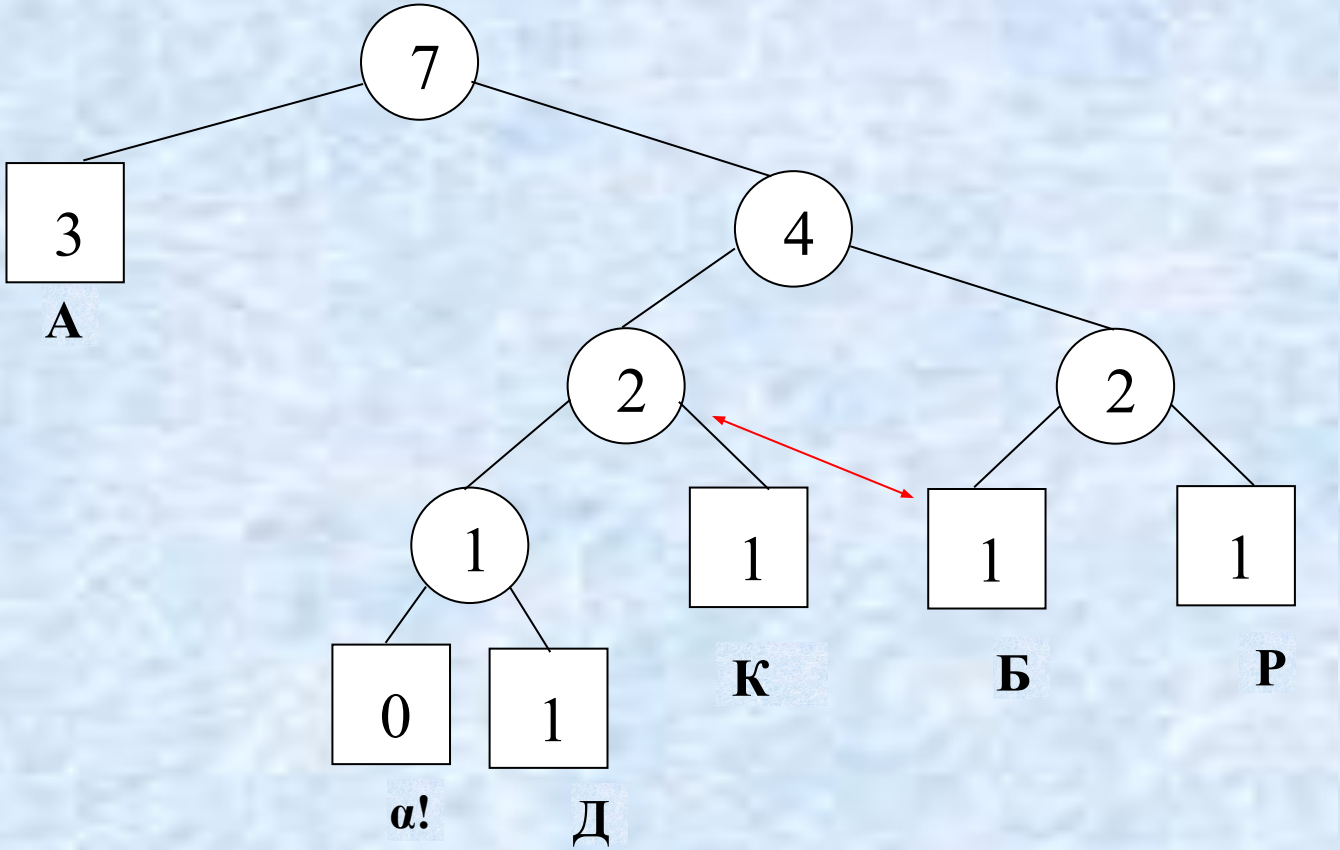
00000000000010001111**0100***01001*01100000
10011011001000011111





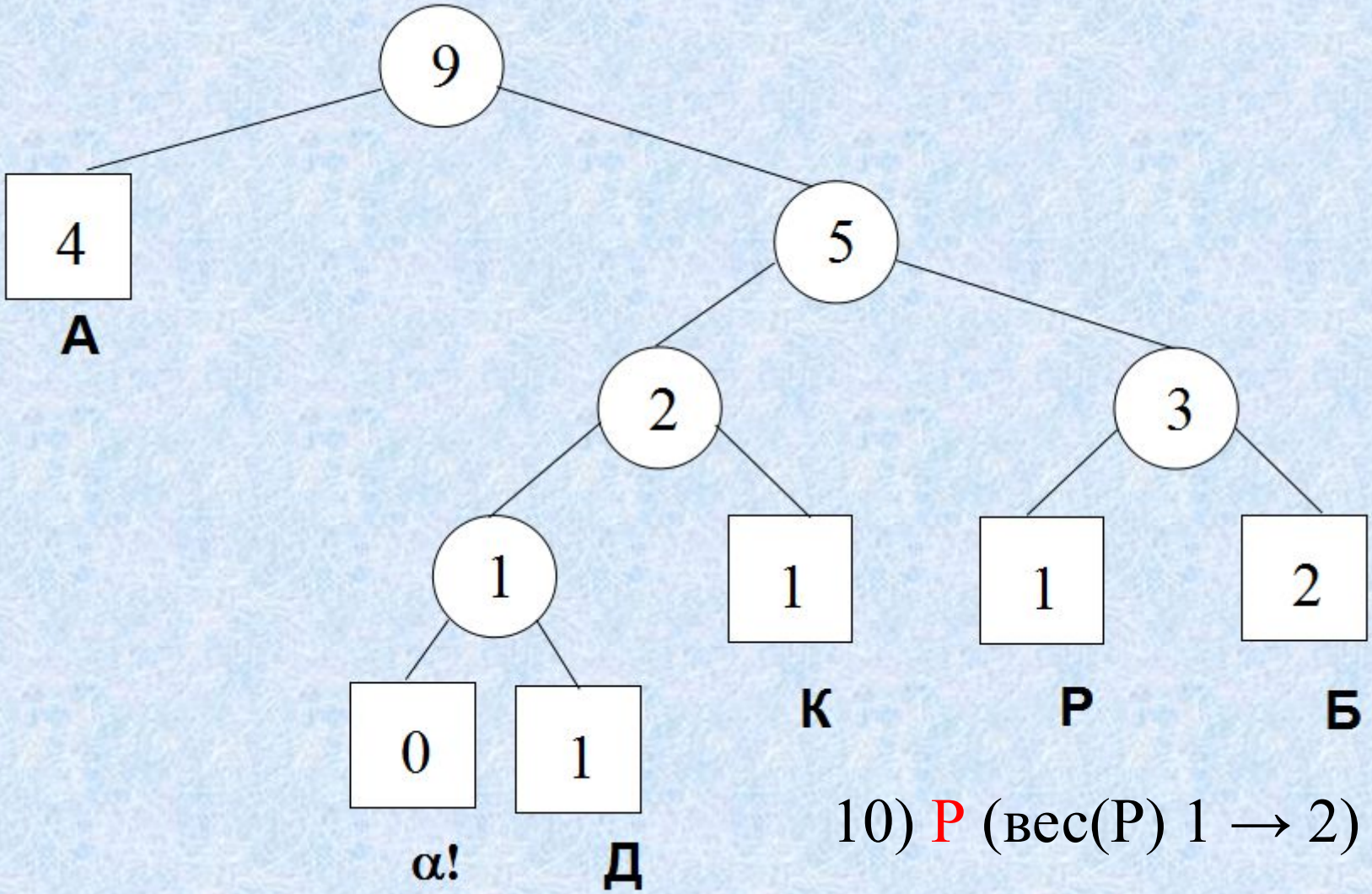
000000000001000111101000100101100000
10011011001000111111

6) А (вес(А) 2→3) 7) α! + Д → АБРАКАД



0000000000001000111101000100101100000
100*110*1100100011111

8) А (вес(А) 3 →4) 9) Б (перевес Б-Р) → АБРАКАДАБ

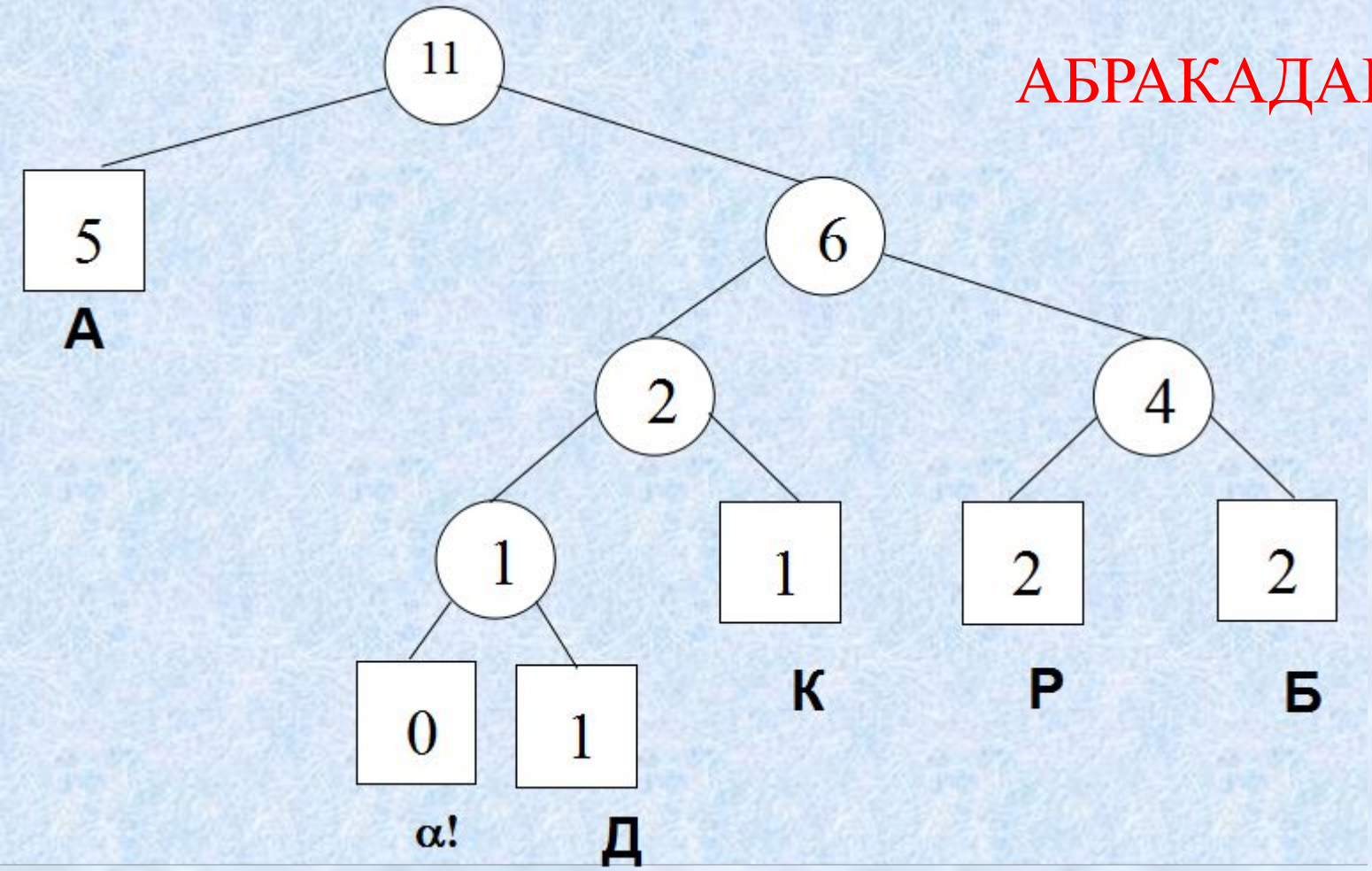


10) Р (вес(Р) 1 → 2)

000000000000100011111010001001011000001001101100100011111

11) А (вес(А) 4 → 5)

АБРАКАДАБРА



12) α! + **!** Конец сообщения

000000000001000111101000100101100000
1001101100**1000***11111*

Проблемы динамического (адаптивного) кодирования Хаффмена

Переполнение:

- переполнение весов

Решение – масштабирование

- длина кода больше размера типа «целое»

Решение - накоплении битов кода в связанном списке, к которому можно добавлять новые узлы

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ