

Алгоритмы и структуры данных

Лекция 8

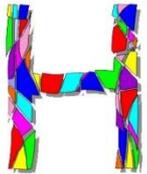
ДЕРЕВЬЯ ХАФФМАНА

(продолжение)

Динамическое кодирование по Хаффману

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

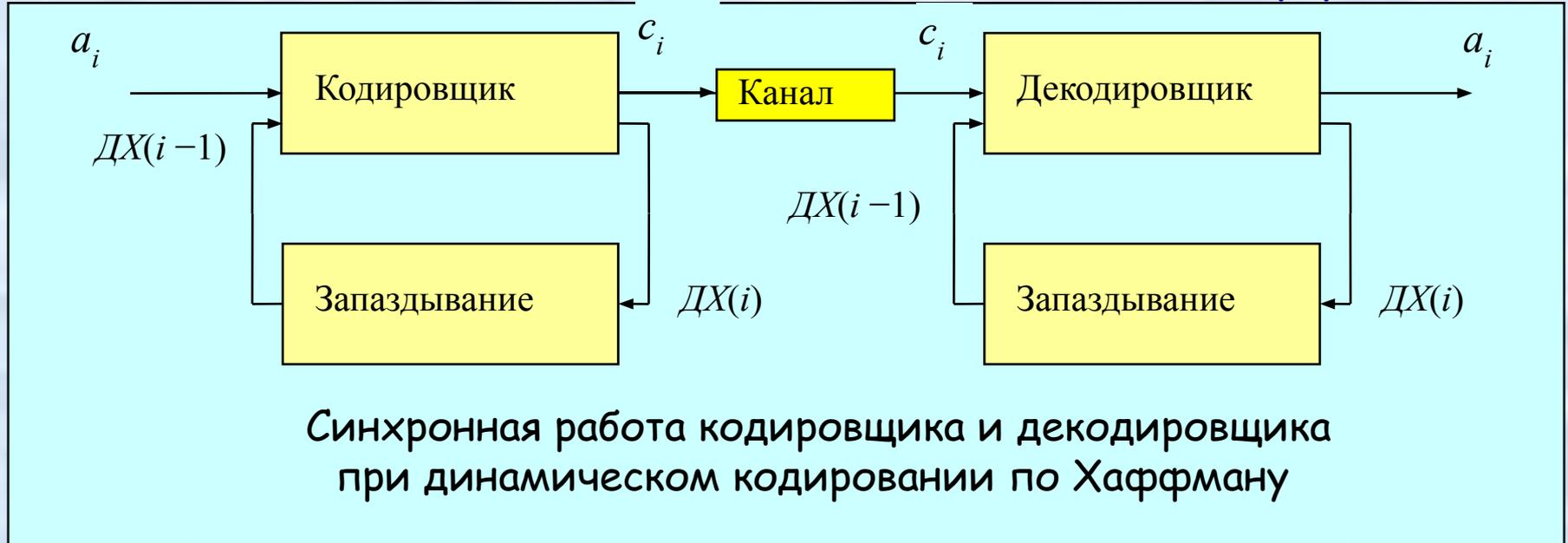
Динамическое кодирование по Хаффману



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



Кодировщик

- 1) построение кода c_i для символа a_i по дереву Хаффмана $DX(i-1)$;
- 2) перестройку $DX(i-1)$ в новое $DX(i)$ с учетом поступившего символа a_i ;

Декодировщик

- 1) восстановление символа a_i по коду c_i и по дереву $DX(i-1)$;
- 2) перестройку $DX(i-1)$ в новое $DX(i)$ с учетом декодированного символа a_i ;

Алгоритм перестроения дерева Хаффмана

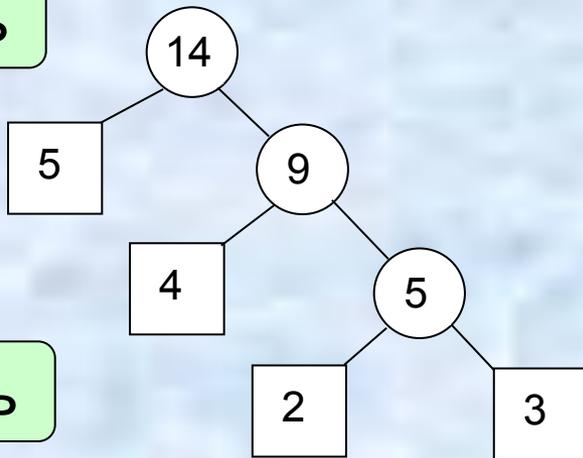
Пусть дерево Хаффмана (ДХ) строится так, что левый сын имеет вес не больший, чем правый.

При этом ДХ, вообще говоря, не единственно.

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

Пояснить

1

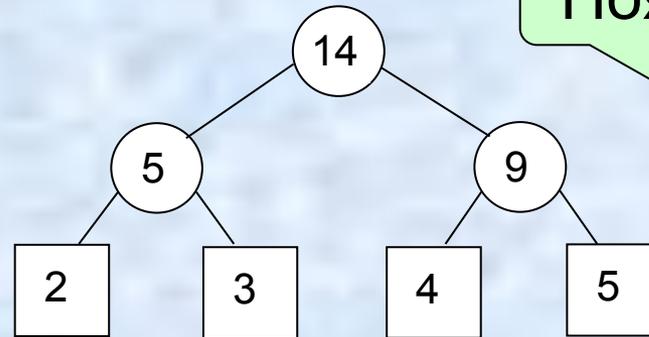


(2, 3, 4, 5, 5, 9, 14)

добавление a_1 ,

$$w_1 \leftarrow w_1 + 1 \quad (6 \leftarrow 5 + 1)$$

Пояснить 1



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

добавление a_4 ,

$$w_4 \leftarrow w_4 + 1 \quad (3 \leftarrow 2 + 1)$$

Пояснить

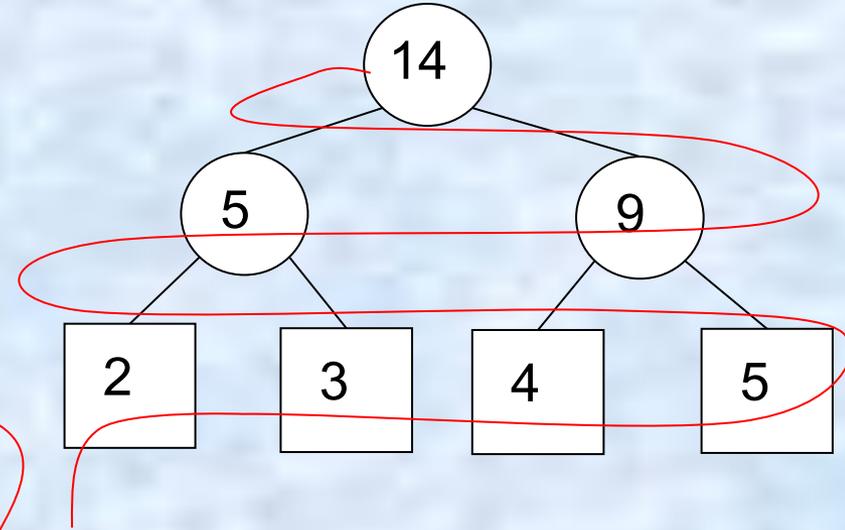
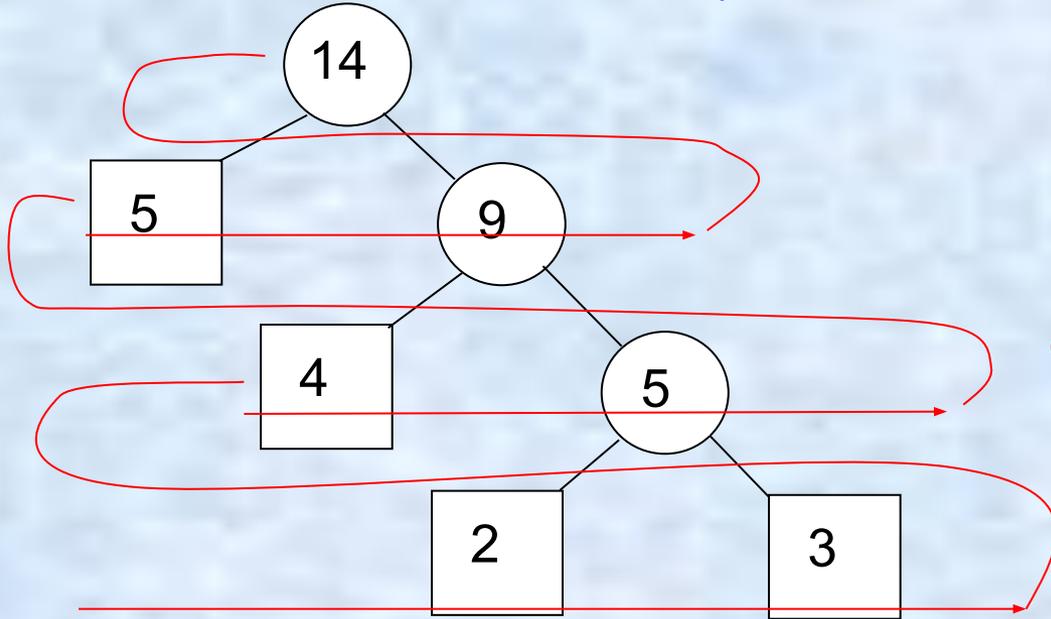
2

Пояснить 2

Пояснения:

левый вариант

правый вариант



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

(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 $x_1 \leq x_2 \leq x_3 \leq \dots \leq x_{2n-1}$, 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

$$w_1 \geq w_2 \geq \dots \geq w_{n-1} \geq w_n.$$

При этом внутренние узлы имеют веса

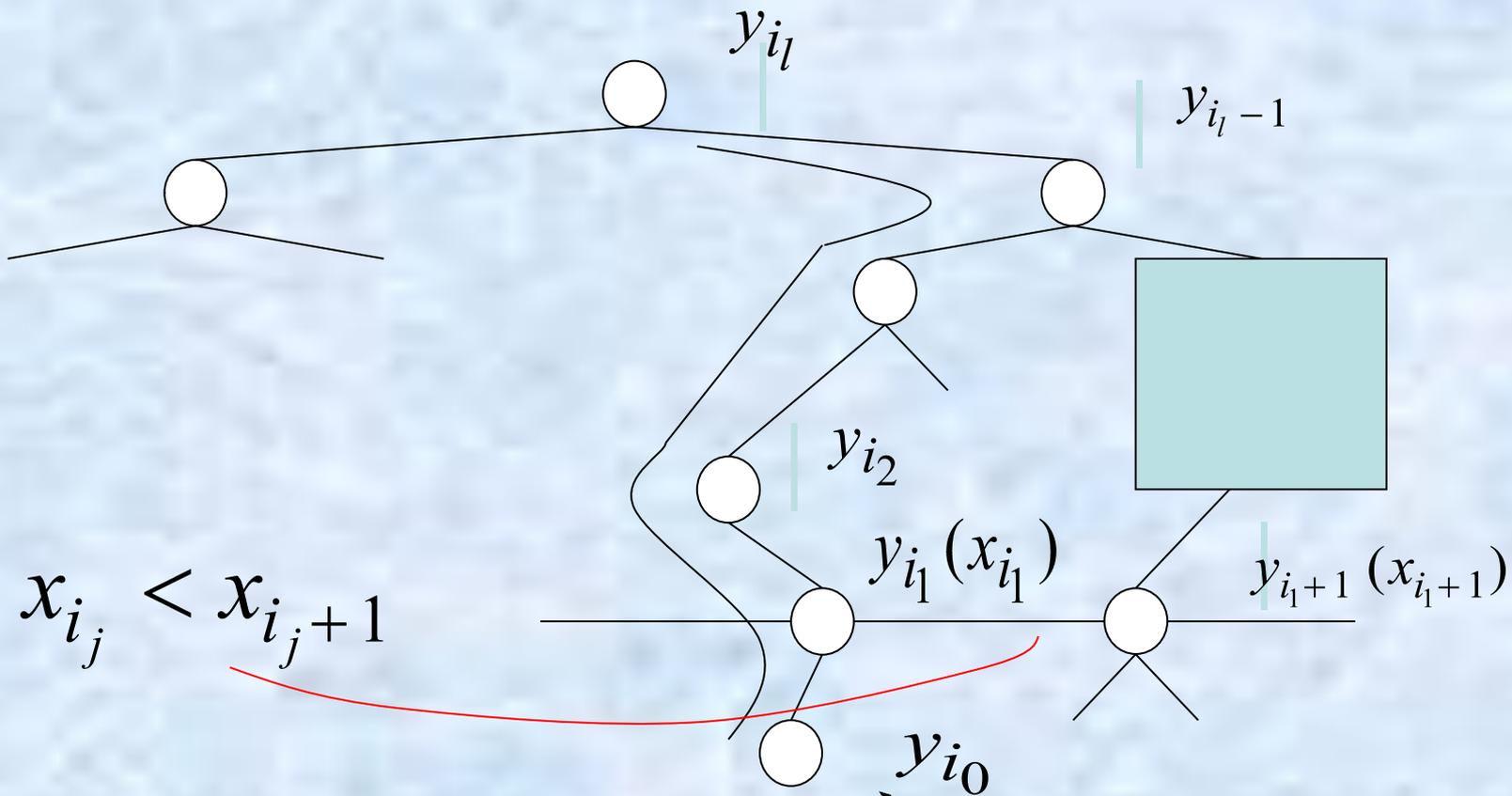
$$x_1 + x_2, x_3 + x_4, \dots, x_{2n-3} + x_{2n-2} = x_{2n-1}.$$

Строго бинарное дерево – упорядоченное, если:

А) n внешних узлов (листьев) получили веса (w_1, w_2, \dots, 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 $(y_1, y_2, \dots, y_{2n-1})$, что если x_i – вес узла y_i , то $x_1 \leq x_2 \leq \dots \leq x_{2n-1}$; узлы y_{2j-1} и y_{2j} – братья (сыновья одного отца).

упорядоченное дерево \Leftrightarrow дерево Хаффмана



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

Пример обеспечения свойства модифицируемости

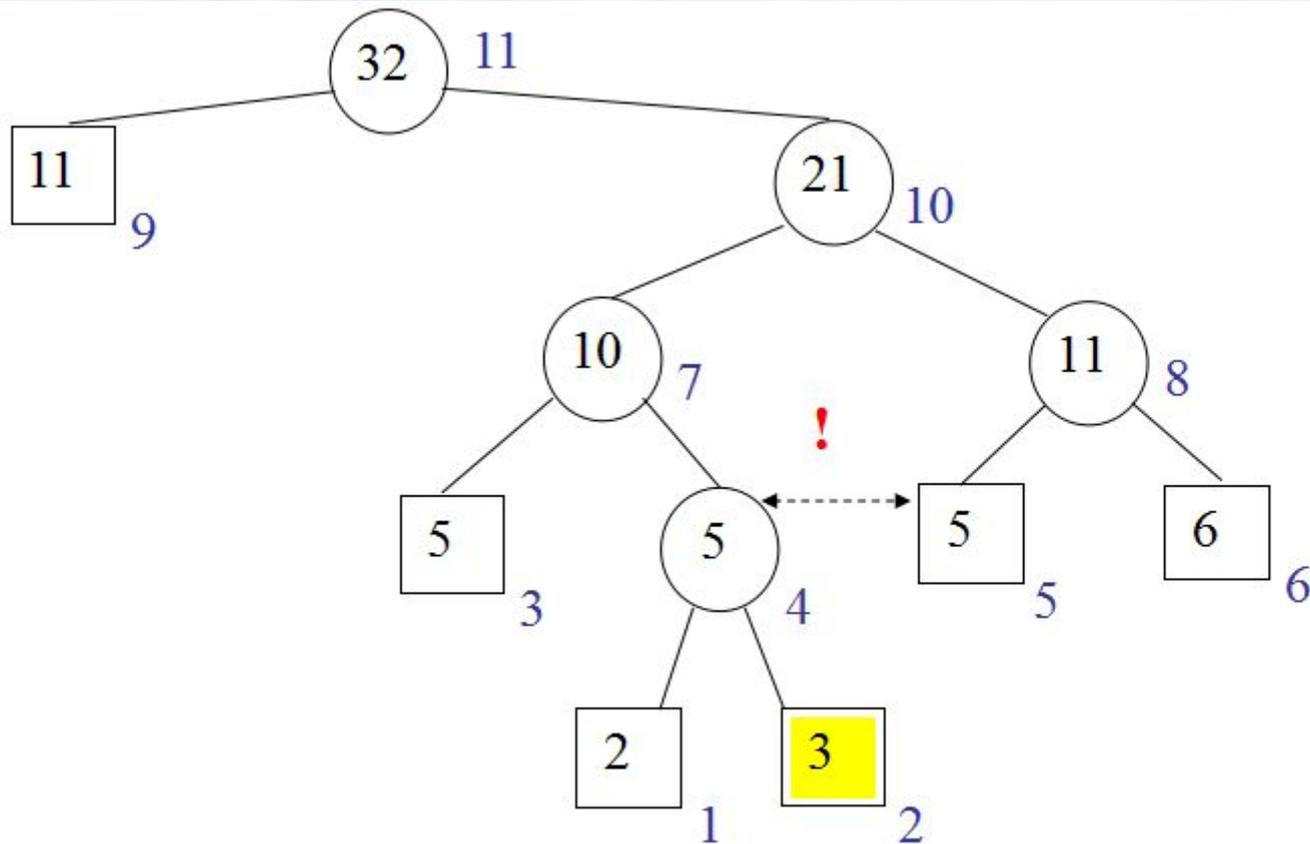
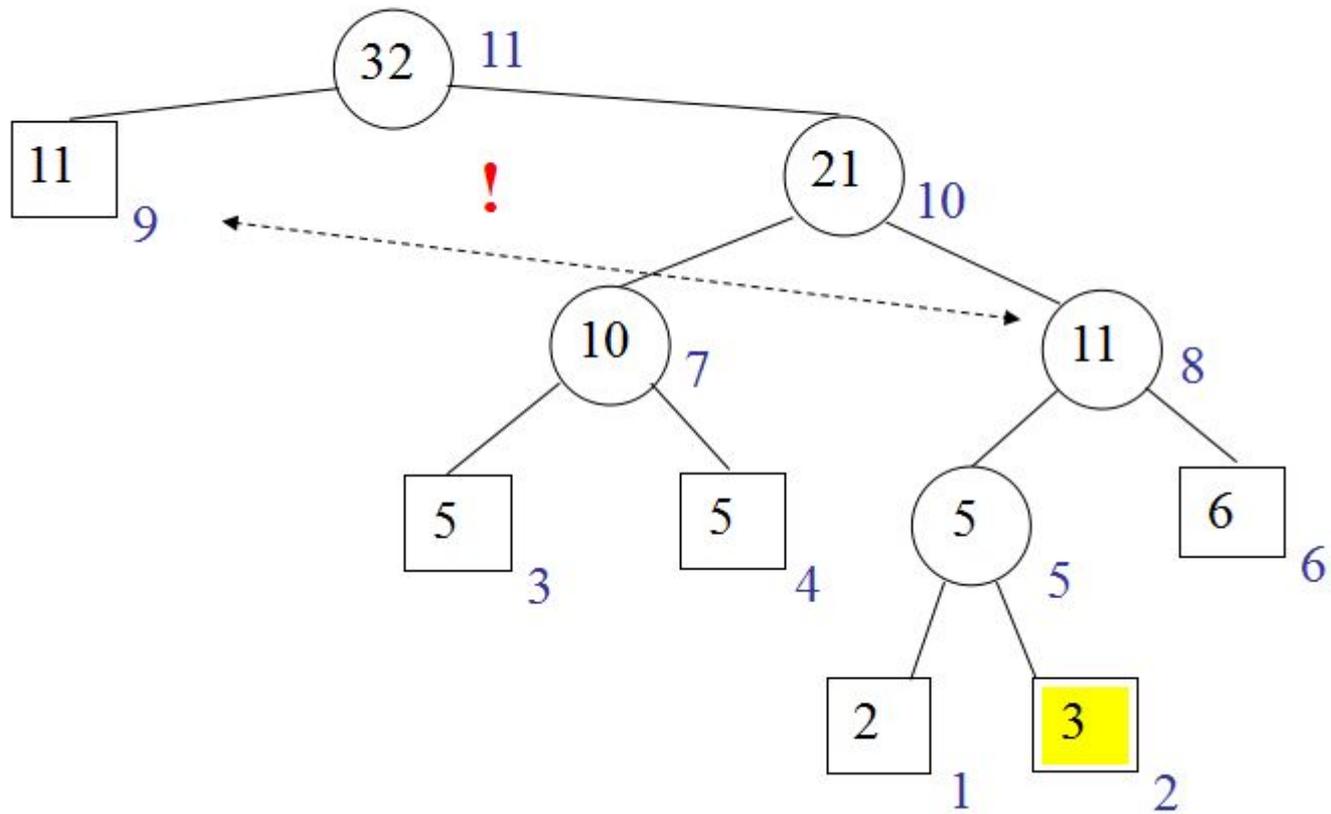
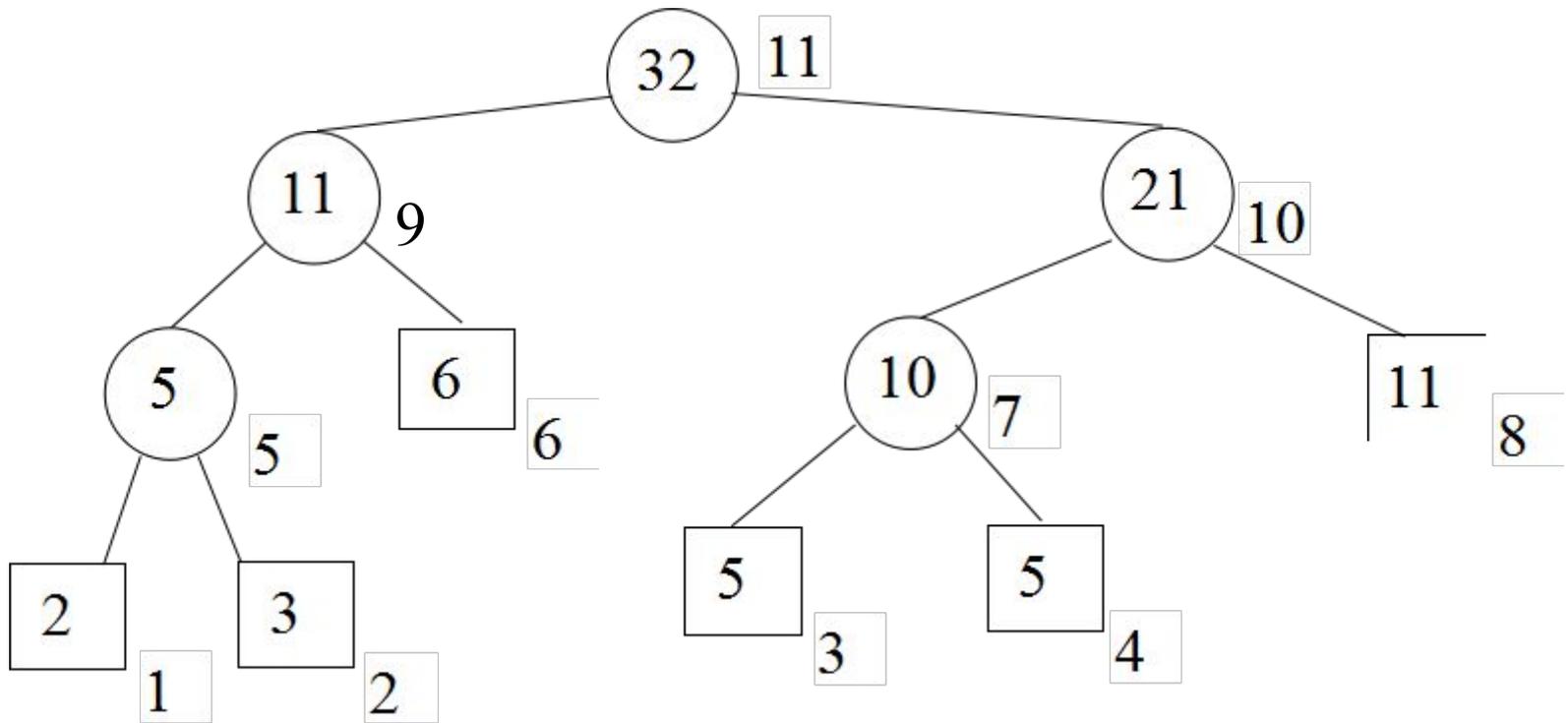


Рис. Пример дерева Хаффмана

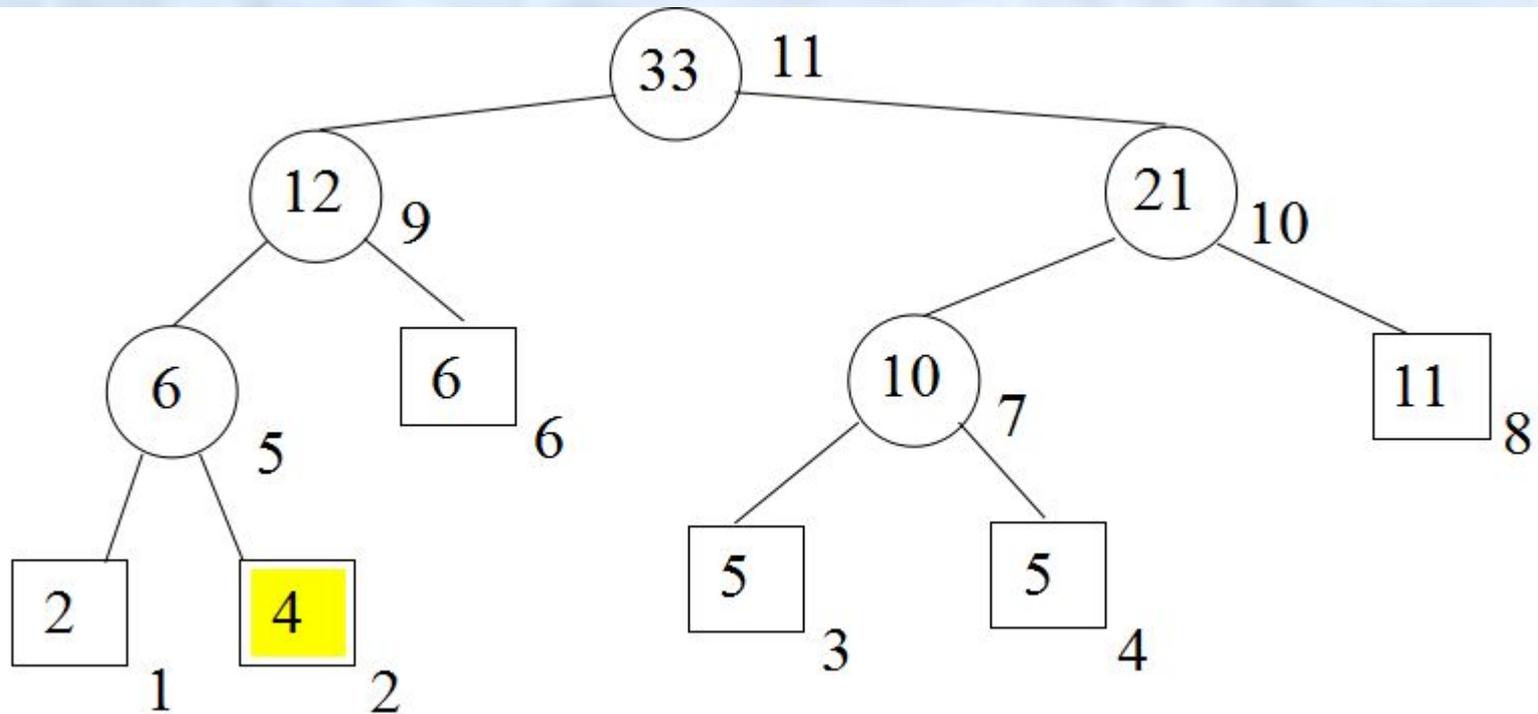
Инвариант: (2, 3, 5, 5, 5, 6, 10, 11, 11, 21, 32),





Это дерево также имеет инвариант $(2, 3, 5, 5, 5, 6, 10, 11, 11, 21, 32)$, и теперь на всем пути $\langle 2, 5, 9, 11 \rangle$ 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

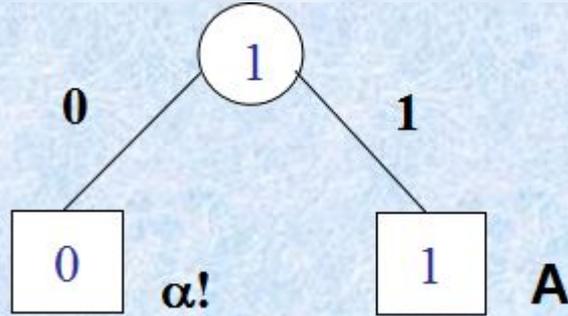
Обозначим $\alpha!$ – узел нулевого веса, отображающий любой символ алфавита, не встречающиеся до сих пор.

Первые вхождения символов кодируются кодом узла $\alpha!$, 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.

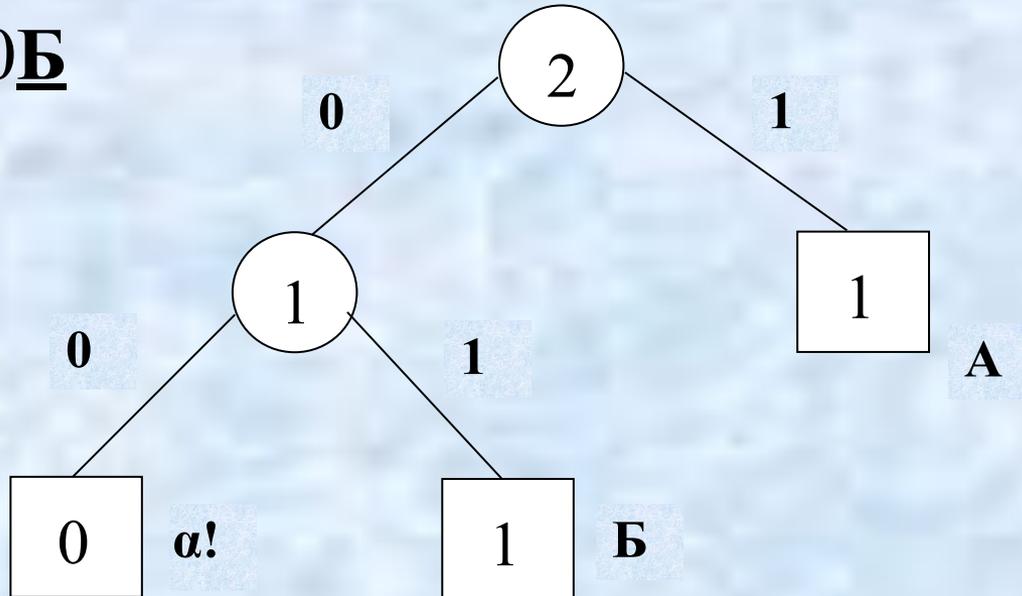
A₁**B**₂**R**₃**A**₄**K**₅**A**₆**D**₇**A**₈**B**₉**R**₁₀**A**₁₁!₁₂

A ⇒ **A**



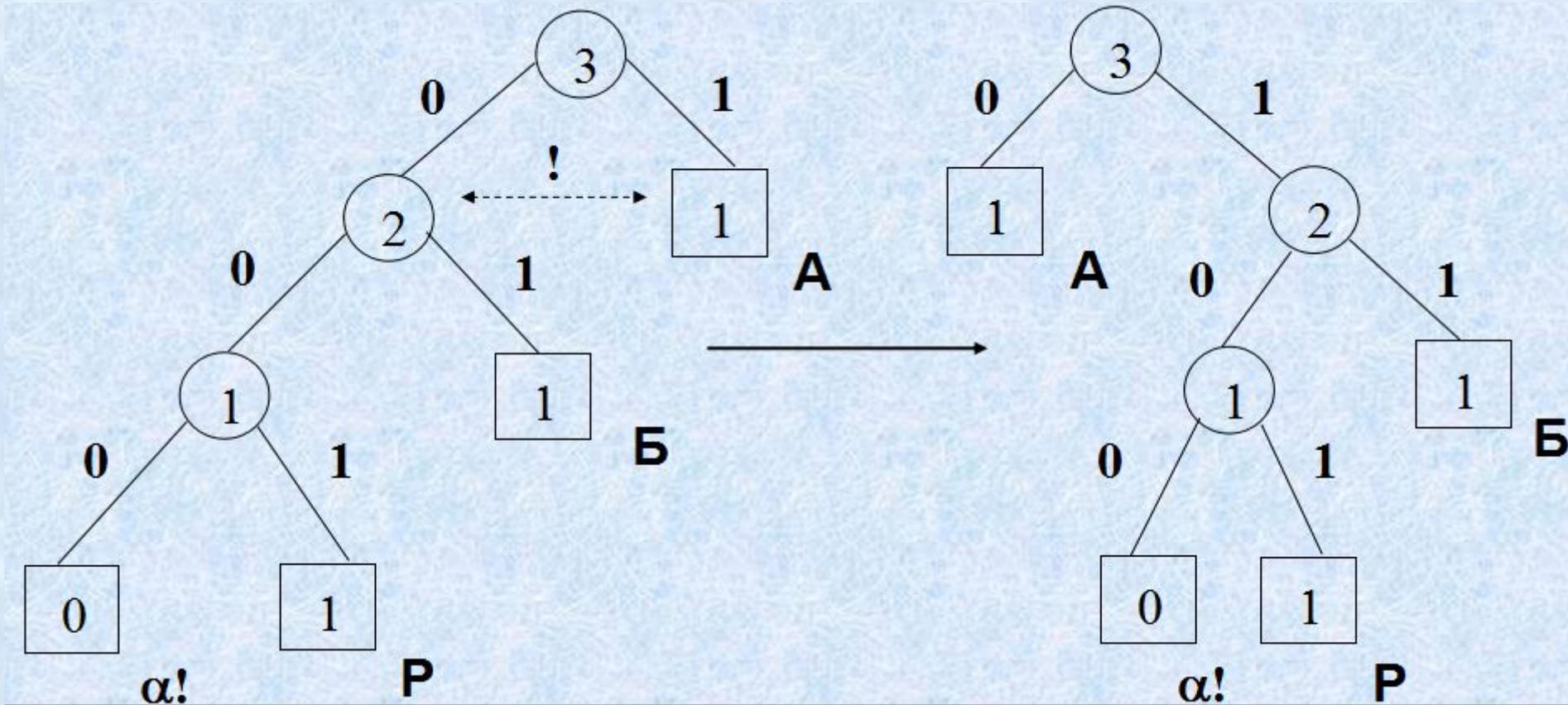
A₁**B**₂**R**₃**A**₄**K**₅**A**₆**D**₇**A**₈**B**₉**R**₁₀**A**₁₁!₁₂

B ⇒ **0B**



A₁ B₂ P₃ A₄ K₅ A₆ Д₇ A₈ B₉ P₁₀ A₁₁!

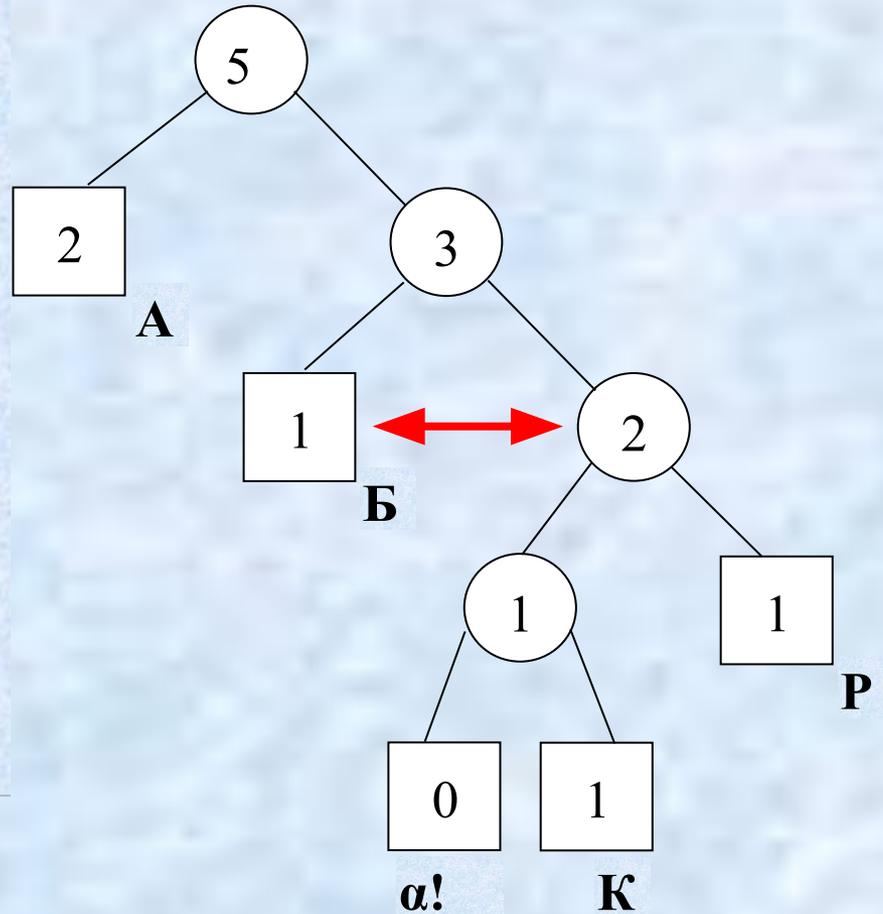
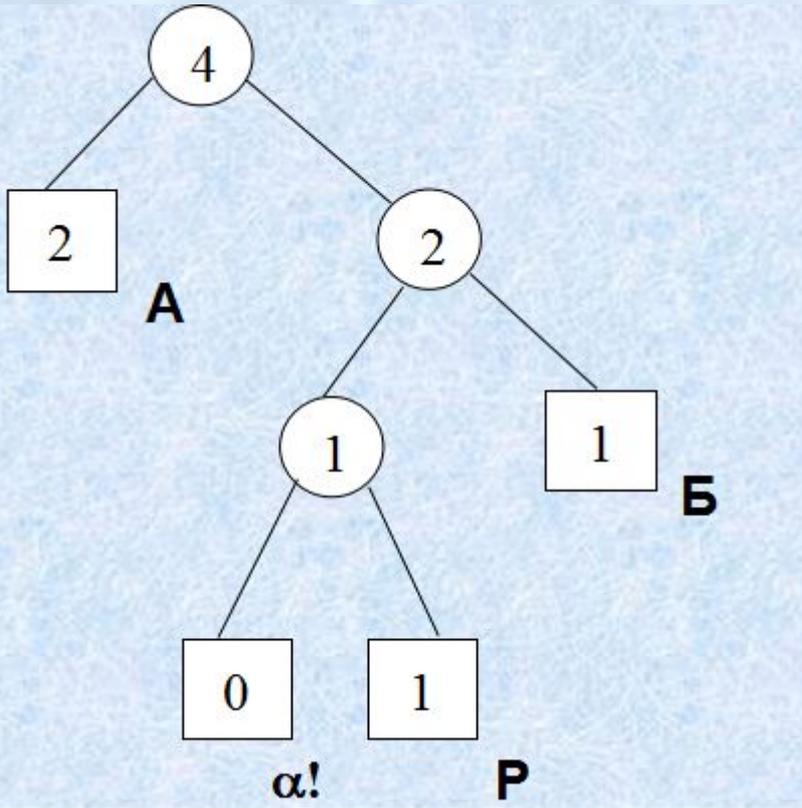
P ⇒ 00P



A₁ **Б**₂ **Р**₃ **A**₄ **К**₅ **A**₆ **Д**₇ **A**₈ **Б**₉ **Р**₁₀ **A**₁₁ **!**₁₂

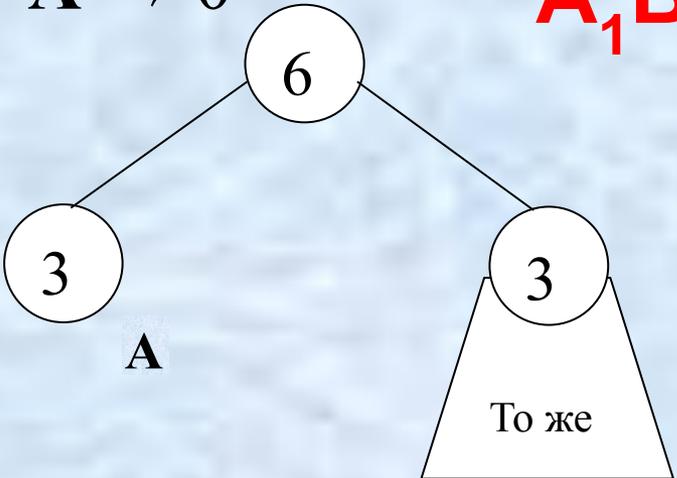
A ⇒ 0

К ⇒ 100 **К**

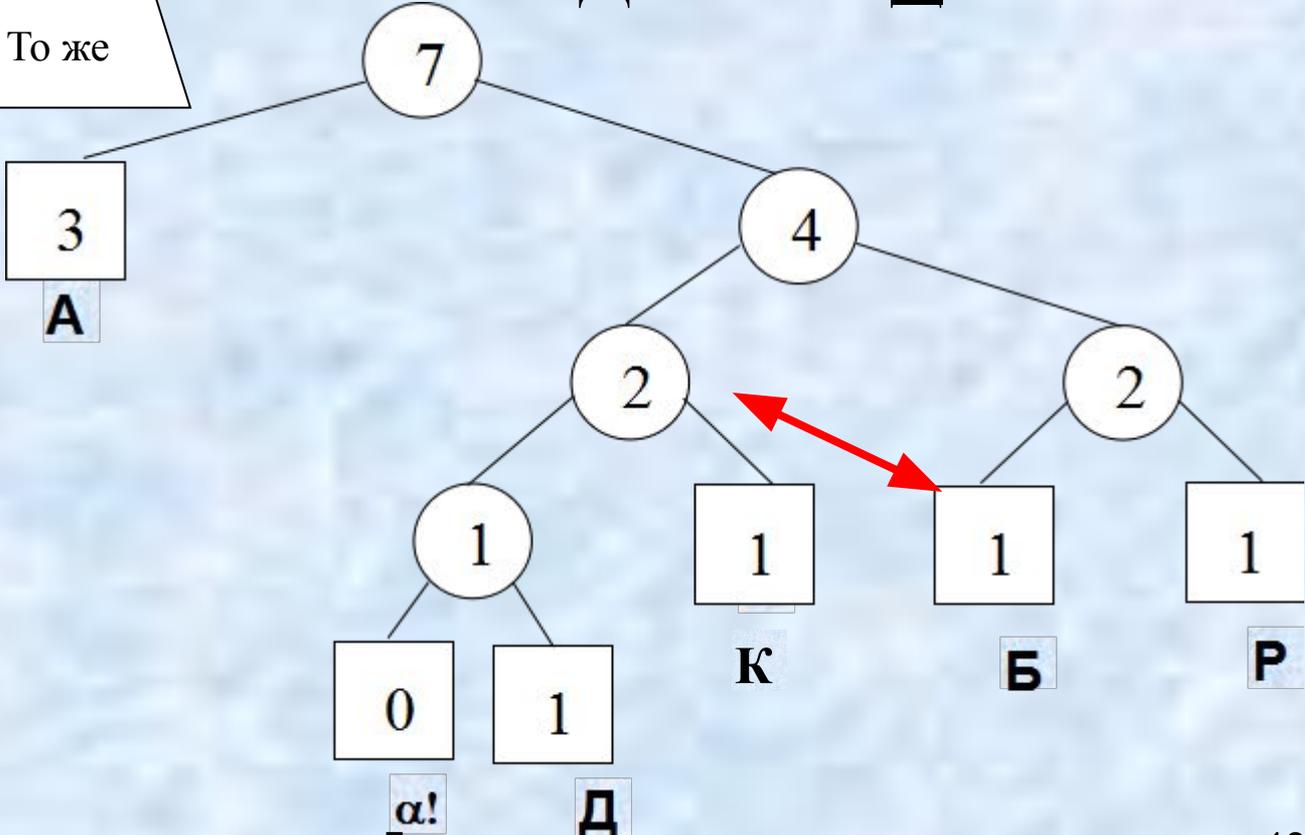


A₁ **Б**₂ **Р**₃ **A**₄ **К**₅ **A**₆ **Д**₇ **A**₈ **Б**₉ **Р**₁₀ **A**₁₁ **!**₁₂

A ⇒ 0



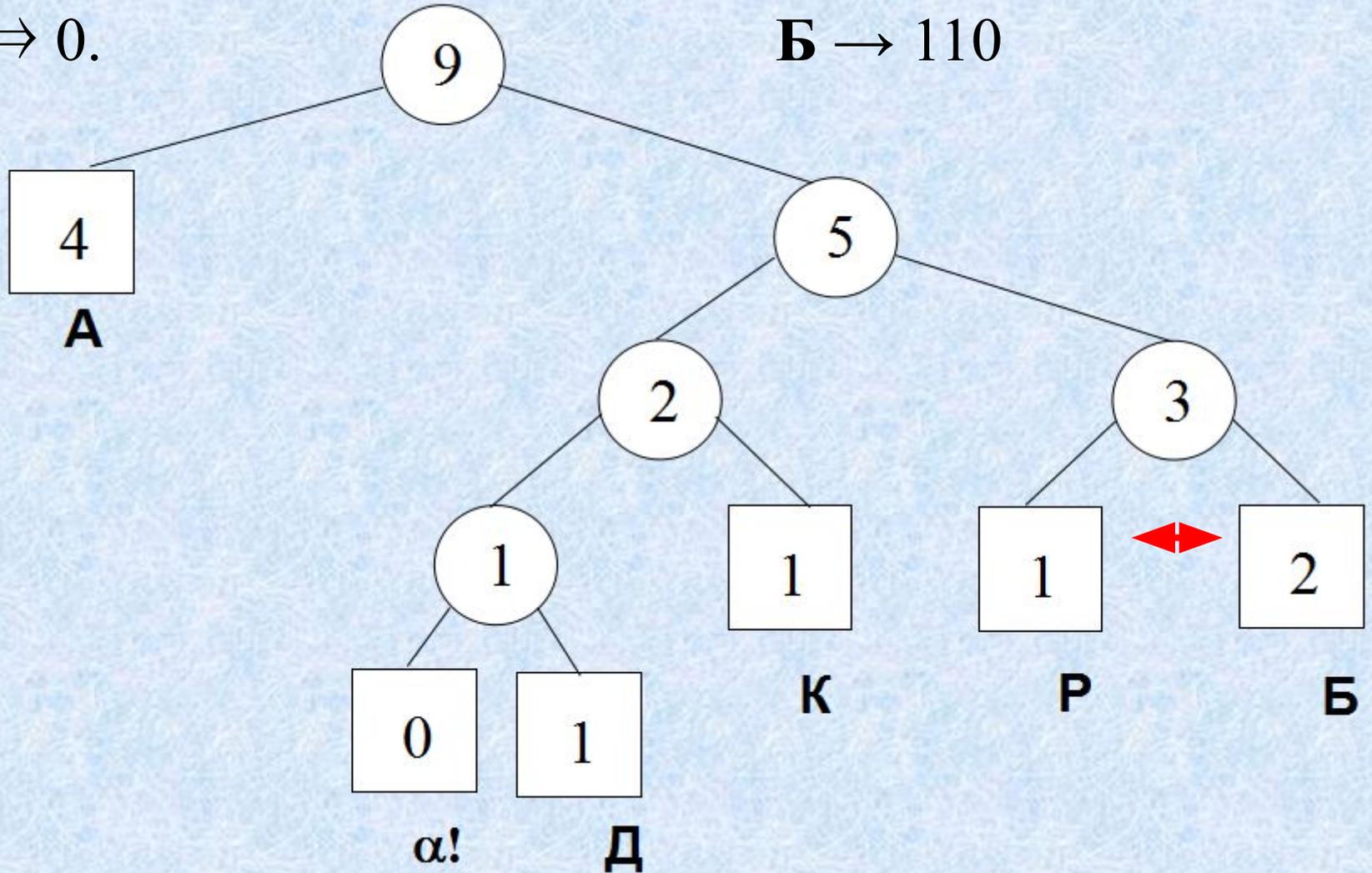
Д ⇒ 1100**Д**



A₁ **B**₂ **P**₃ **A**₄ **K**₅ **A**₆ **Д**₇ **A**₈ **Б**₉ **P**₁₀ **A**₁₁ **!**₁₂

A ⇒ 0.

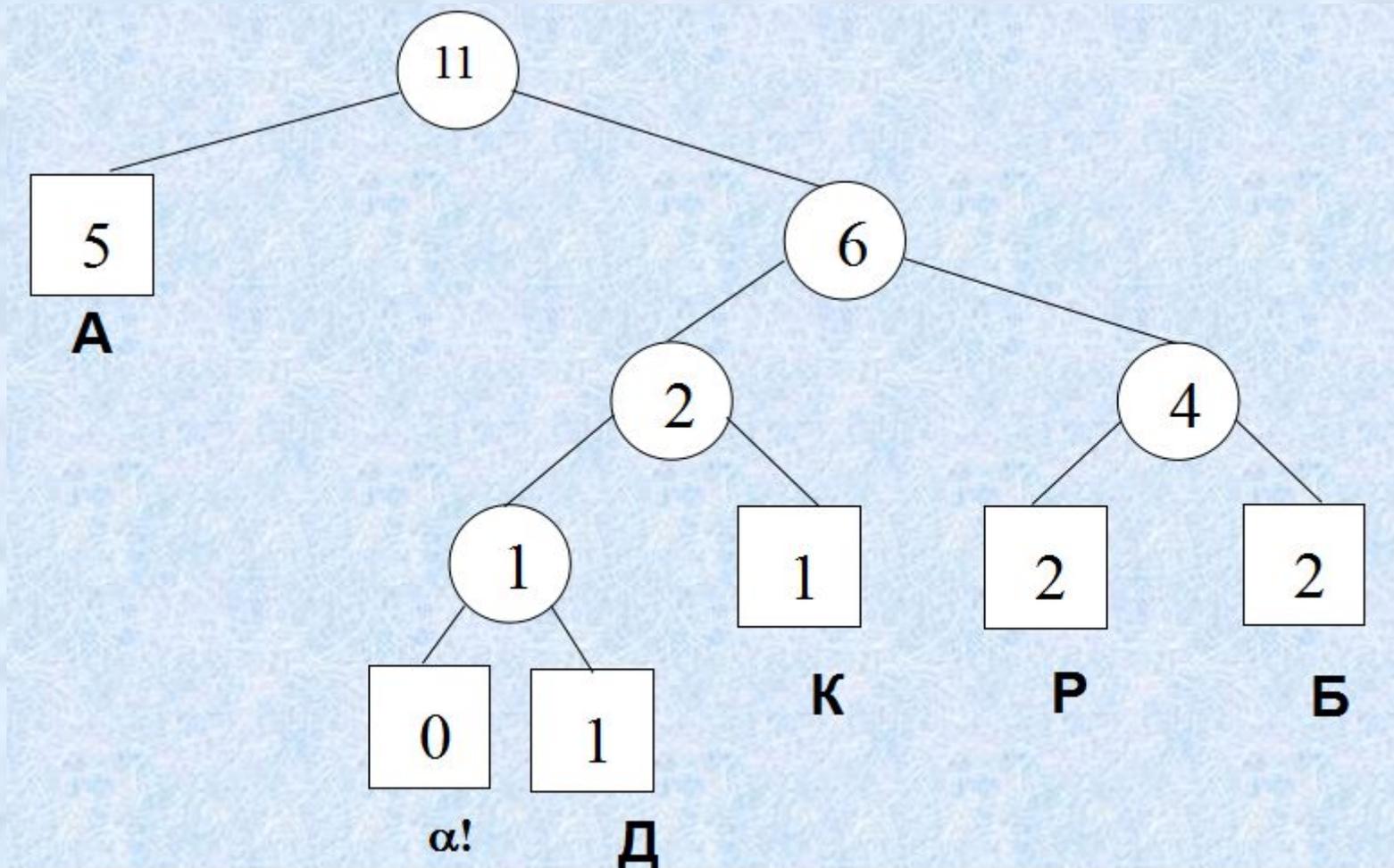
Б → 110



A₁ B₂ P₃ A₄ K₅ A₆ Д₇ A₈ Б₉ P₁₀ A₁₁!₁₂

P ⇒ 110 (ср. шаг 9 !)

A ⇒ 0



A₁ B₂ P₃ A₄ K₅ A₆ Д₇ A₈ B₉ P₁₀ A₁₁ !₁₂

! ⇒ 1000!

Конец текста.

Перестраивать дерево не требуется.

A 0 B 00 P 0 100 K 0 1100 Д 0 110 110 0 1000!

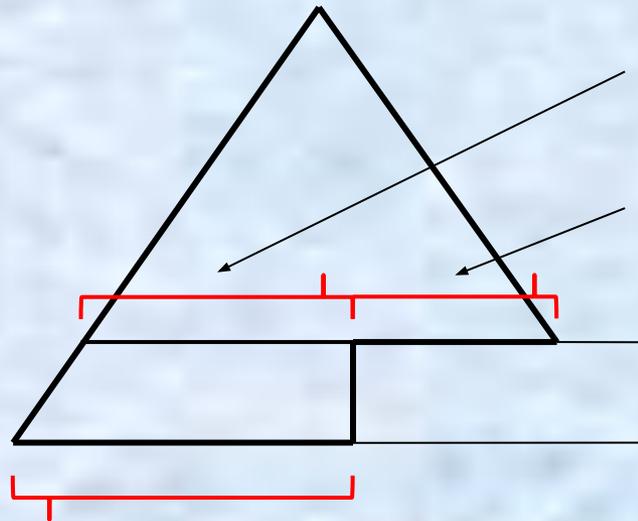
↑ ↑ ↑ ↑ ↑ ↑

A A AB P A

Кодирование первых вхождений

$$n = 2^p + q$$

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



q внутренних узлов
(отцов $2q$ листьев)

$2^p - q = n - 2q$ листьев

p -й уровень

$(p + 1)$ -й уровень

$2q$ листьев

Если $1 \leq i \leq 2q$, 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

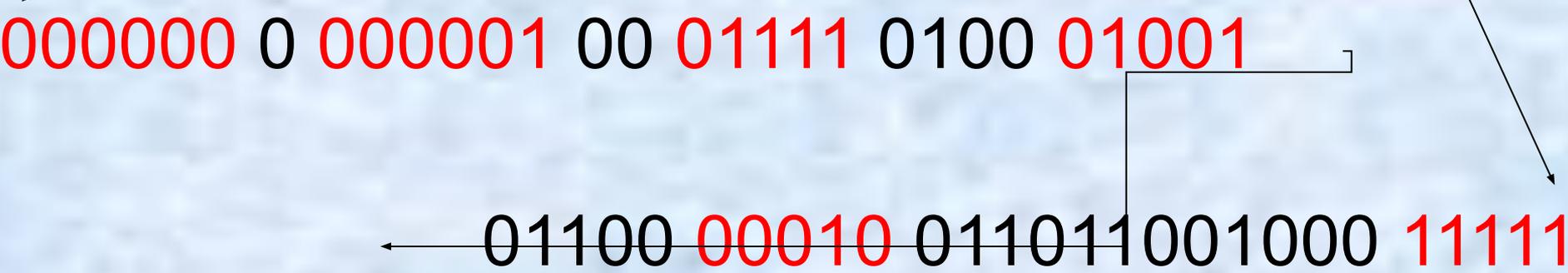
код (**Д**) = 00010

код (**К**) = 01001

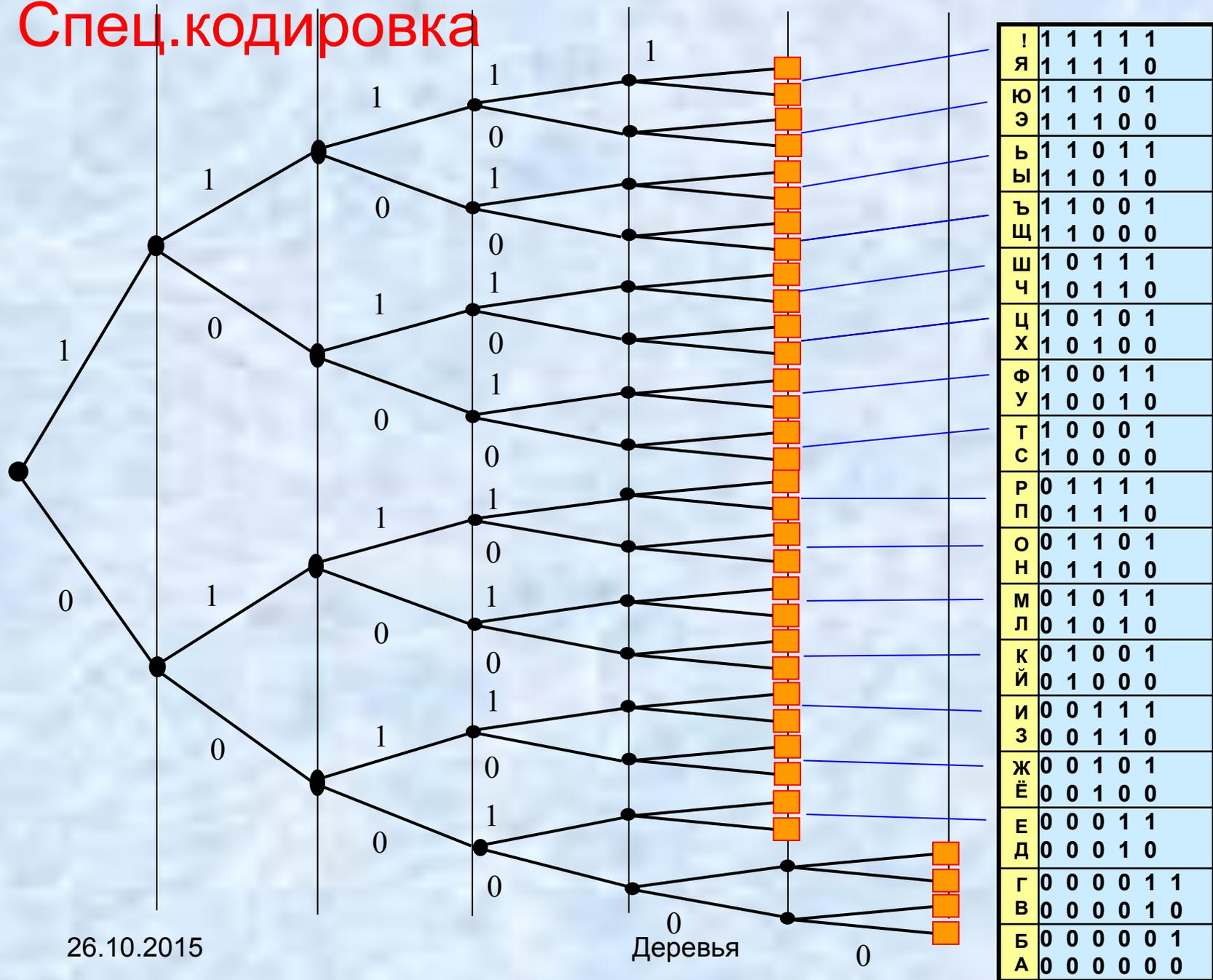
код (**Р**) = 01111 код (!)
= 11111

A0B00R0100K01100D011011001000!

A0B00R0100K 01100D011011001000!



Спец.кодировка

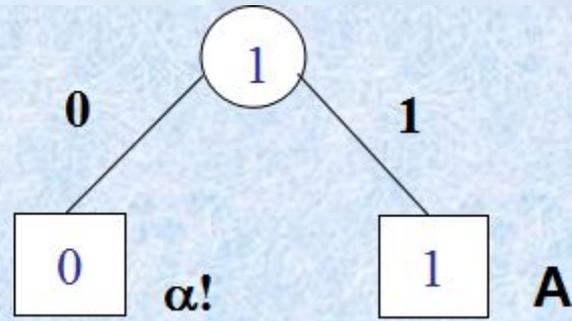


Декодирование

0000000000001000111101000100101100000
1001101100100011111

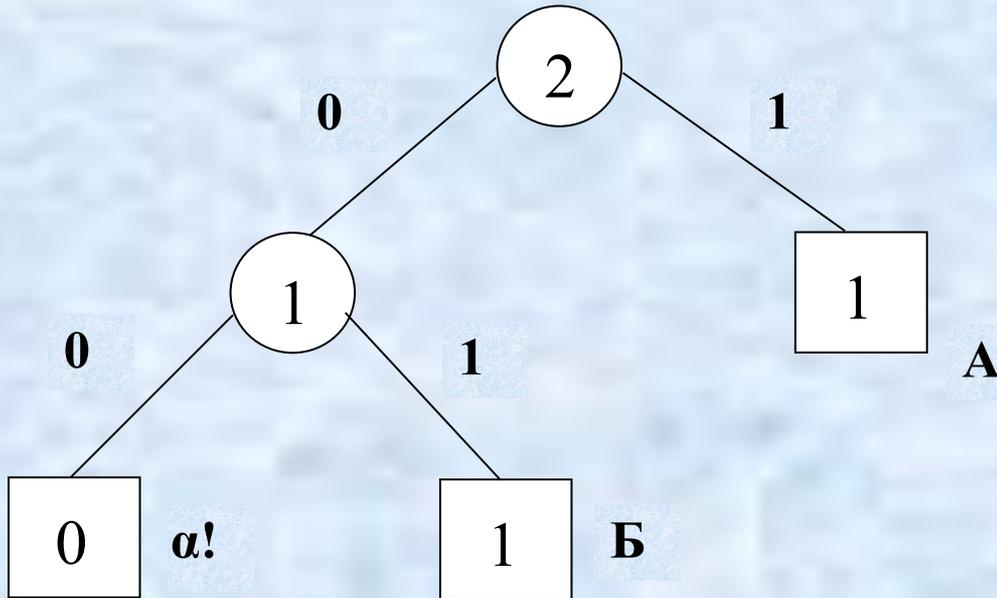
1) В спец. кодировке: 000000 - A

0000000000001000111101000100101100000
1001101100100011111



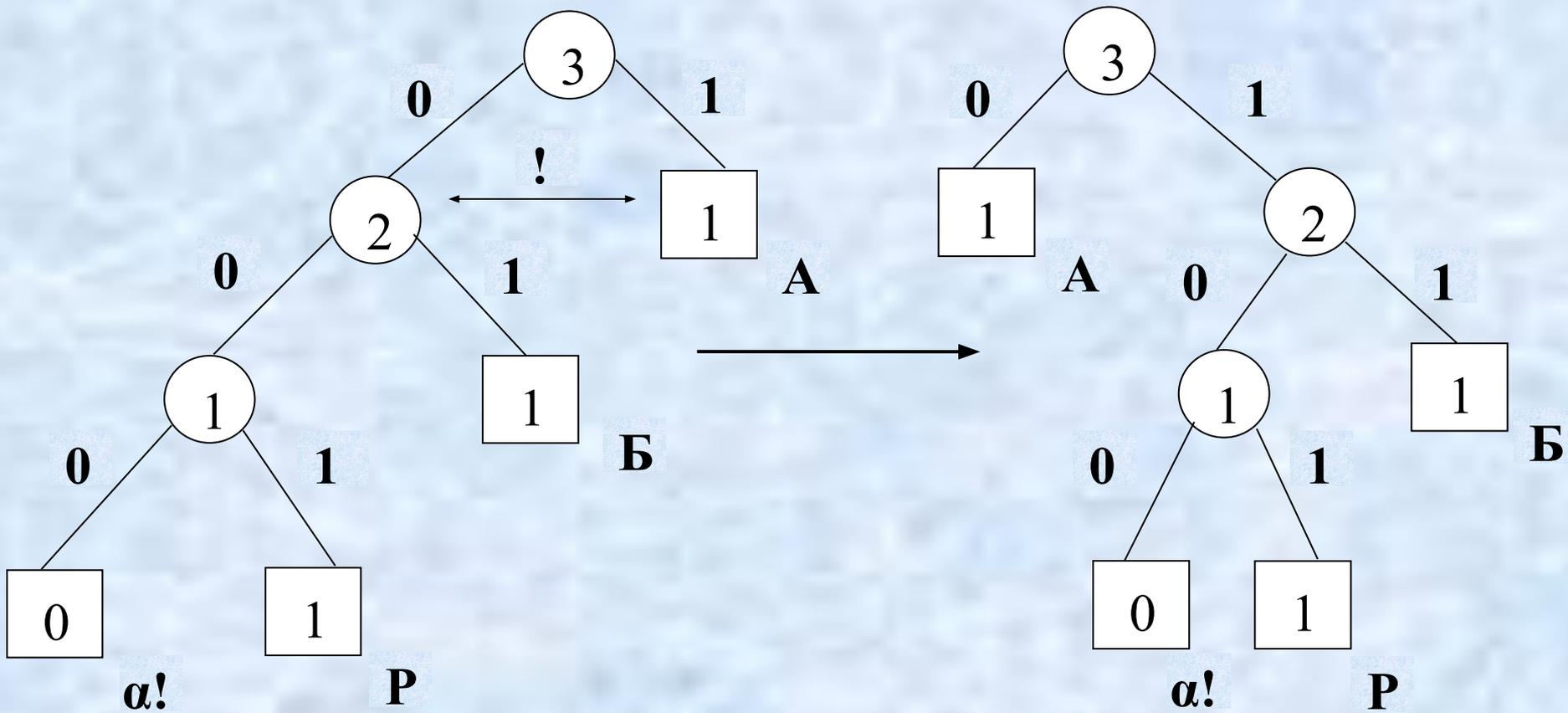
2) $\alpha! + Б$

00000000000001000111101000100101100000
1001101100100011111



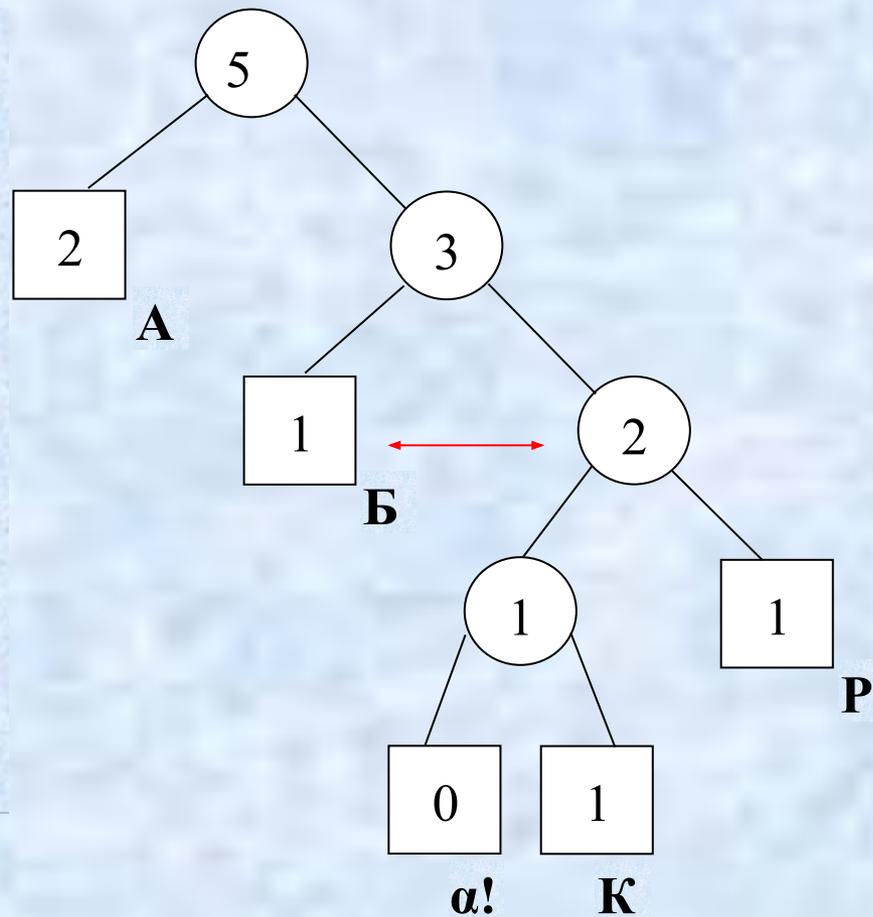
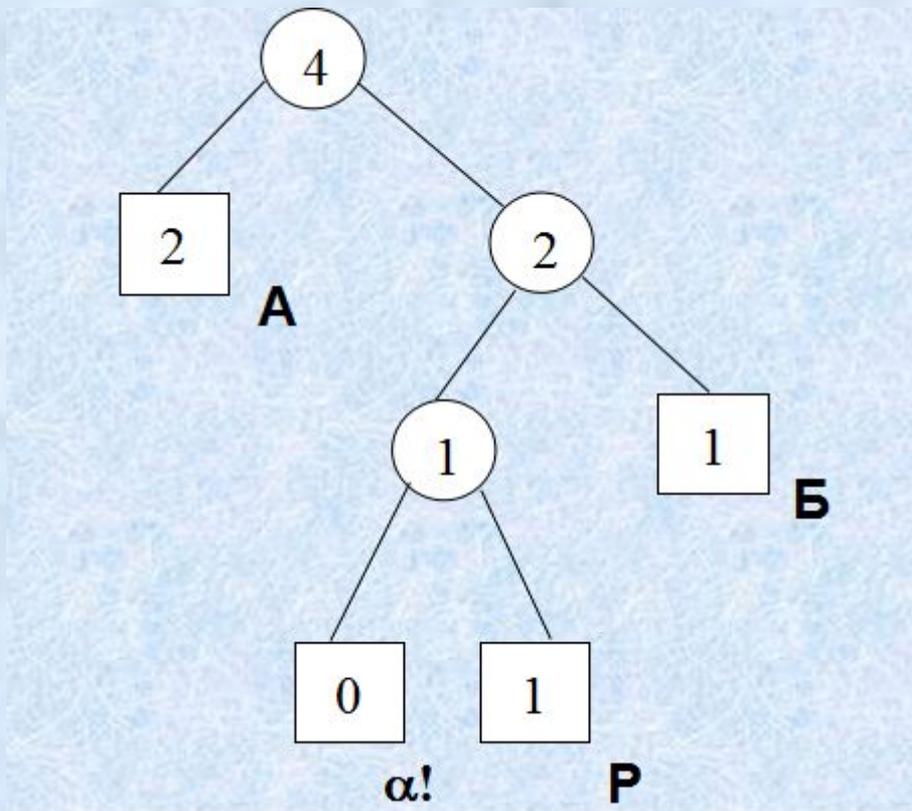
3) $\alpha! + P \rightarrow АБР$

00000000000001000111101000100101100000
1001101100100011111



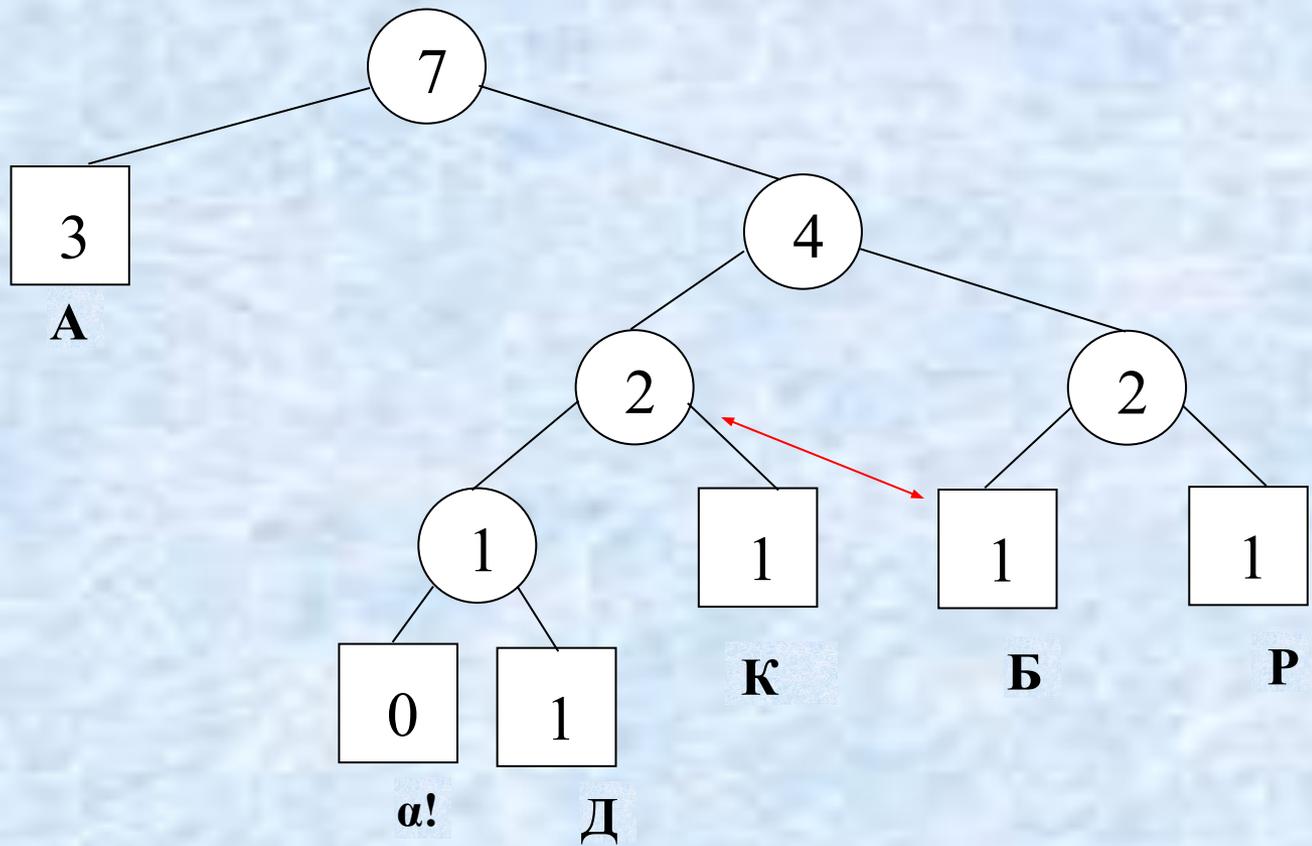
4) A (вес(A) $1 \rightarrow 2$) 5) $\alpha! + K$

0000000000001000111101000100101100000
 100110110010001111



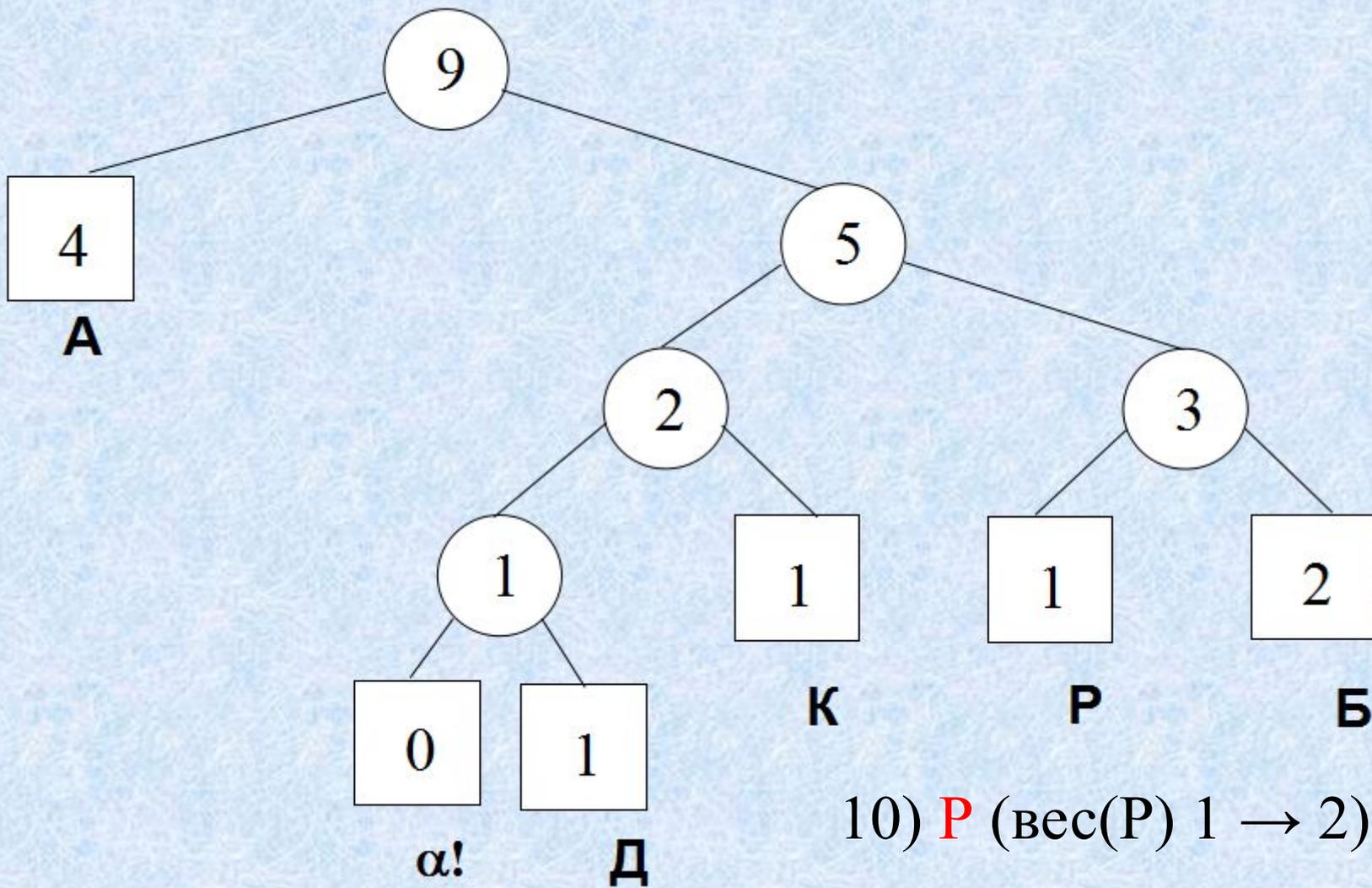
00000000000001000111101000100101100000
 1001101100100011111

6) **А** (вес(А) $2 \rightarrow 3$) 7) **α! + Д \rightarrow АБРАКАД**



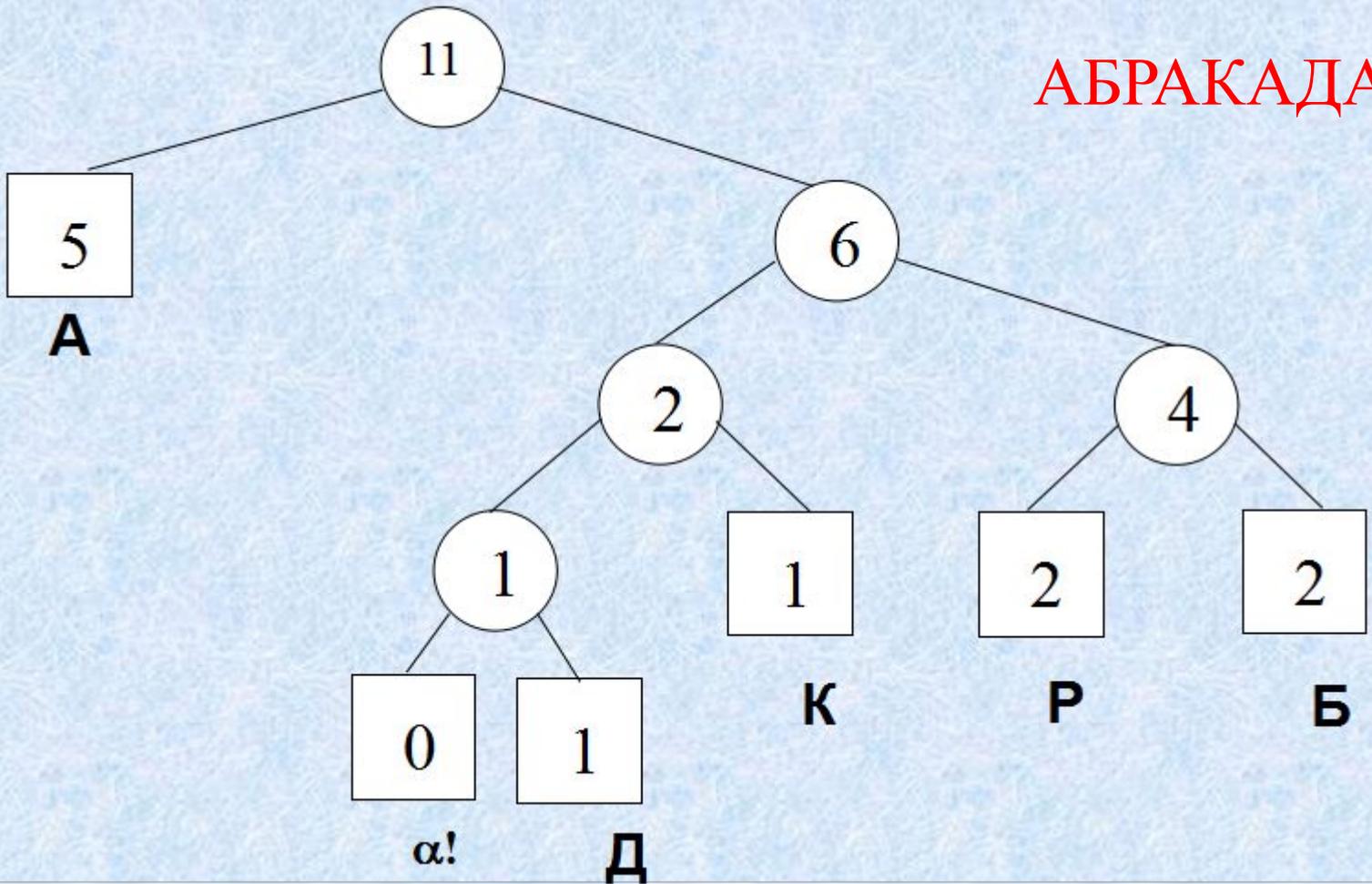
0000000000001000111101000100101100000
 1001101100100011111

8) **A** (вес(A) 3 \rightarrow 4) 9) **Б** (перевес Б-Р) \rightarrow **АБРАКАДАБ**



0000000000001000111101000100101100000
 100110**1100**100011111

АБРАКАДАБРА



12) $\alpha! + !$ Конец сообщения

0000000000001000111101000100101100000
1001101100**100011111**

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