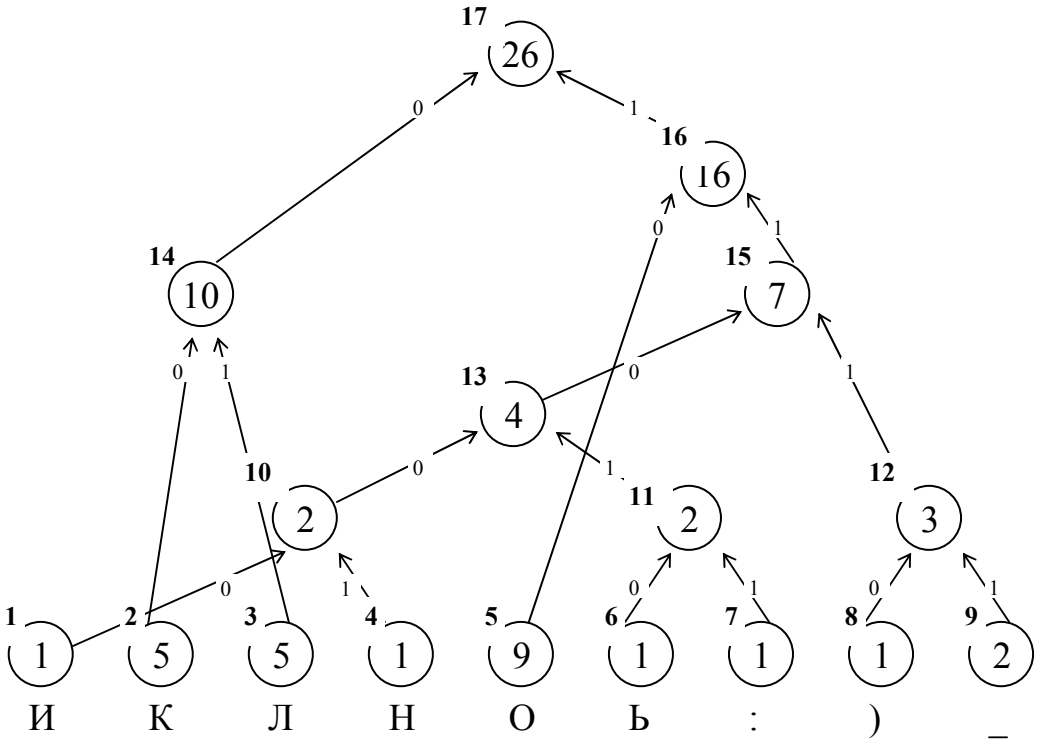
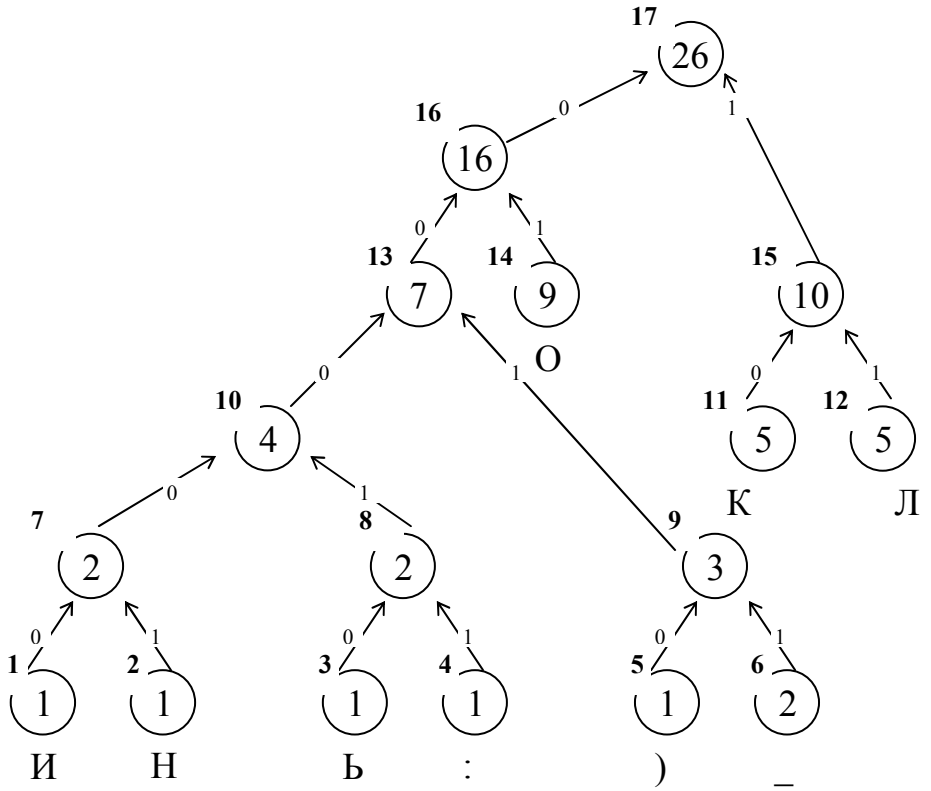
Алгоритм сжатия Хаффмана (динамический)

Неупорядоченное дерево Хаффмана



Упорядоченное дерево Хаффмана



Исходные коды постоянной длины m (4 бита)

И	0	0	0	0	**Н**	0	0	1	1	**:**	0	1	1	0
К	0	0	0	1	**О**	0	1	0	0	**)**	0	1	1	1
Л	0	0	1	0	**Ь**	0	1	0	1	**_**	1	0	0	0

Исходный текст: КОЛОКОЛ_ОКОЛО_КОЛОКОЛЬНИ:)

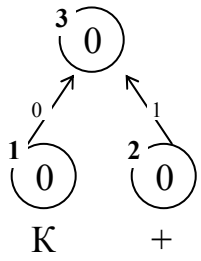
Создание динамического кодового дерева и адаптивного кода Хаффмана

Итерация 0. Создание пустого дерева.

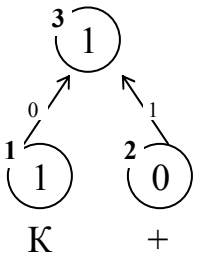
Итерация 1. Добавление символа "К"

Проверяем наличие символа "К" в текущем дереве и формируем код 0001 (исходный код символа "К")

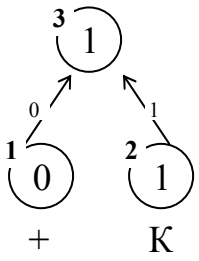
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов



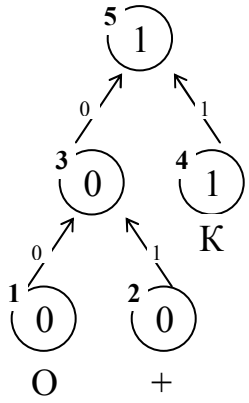
Упорядочивание дерева



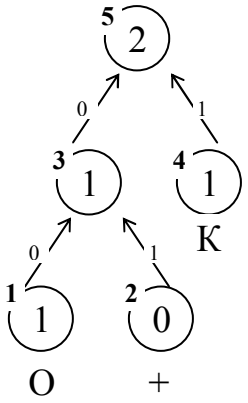
Итерация 2. Добавление символа "О"

Проверяем наличие символа "О" в текущем дереве и формируем код 0 0100 (код символа "+" в текущем дереве, полученном после итерации 1 и исходный код символа "О")

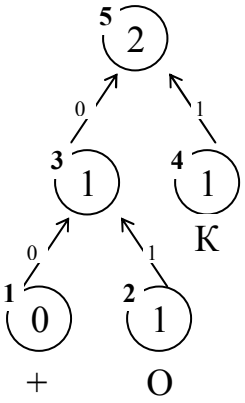
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов



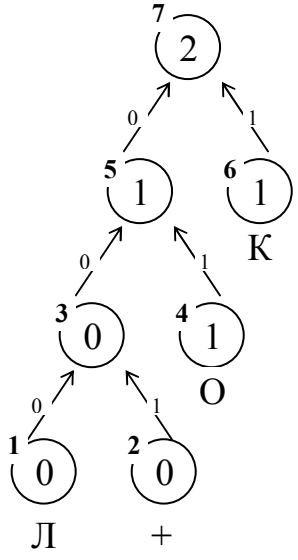
Упорядочивание дерева



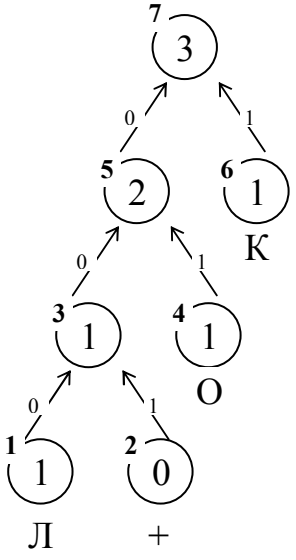
Итерация 3. Добавление символа "Л"

Проверяем наличие символа "Л" в текущем дереве и формируем код 00 0010(код символа "+" в текущем дереве, полученном после итерации 2 и исходный код символа "Л")

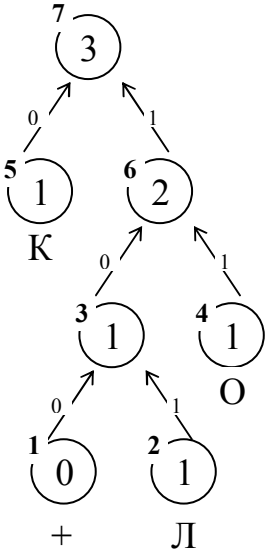
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов



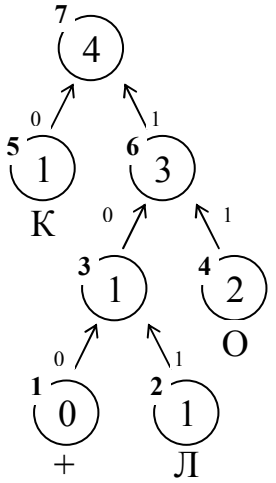
Упорядочивание дерева



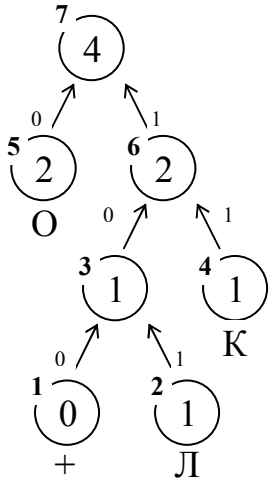
Итерация 4. Добавление символа "О"

Проверяем наличие символа "О" в текущем дереве и формируем код 11 (код символа "О" в текущем дереве, полученном после итерации 3)

Увеличение веса символа и вышестоящих узлов



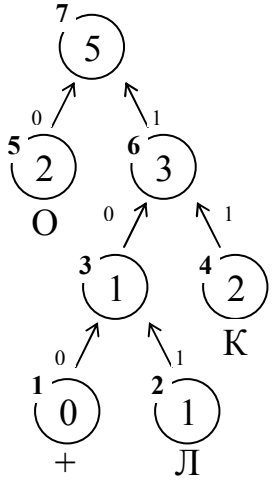
Упорядочивание дерева



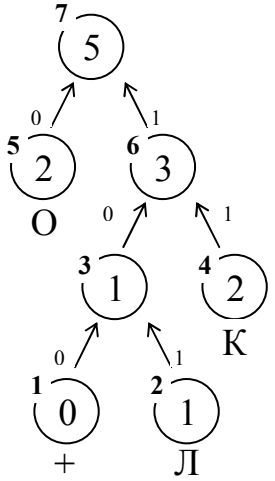
Итерация 5. Добавление символа "К"

Проверяем наличие символа "К" в текущем дереве и формируем код 11 (код символа "К" в текущем дереве, полученном после итерации 4)

Увеличение веса символа и вышестоящих узлов



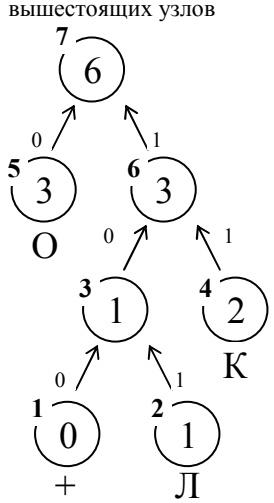
Упорядочивание дерева



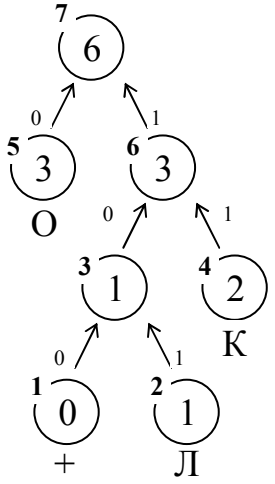
Итерация 6. Добавление символа "О"

Проверяем наличие символа "О" в текущем дереве и формируем код 0 (код символа "О" в текущем дереве, полученном после итерации 5)

Увеличение веса символа и вышестоящих узлов



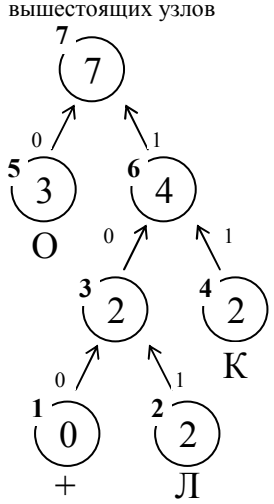
Упорядочивание дерева



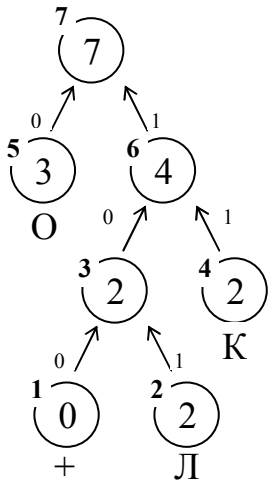
Итерация 7. Добавление символа "Л"

Проверяем наличие символа "Л" в дереве и формируем код 101 (код символа "Л" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



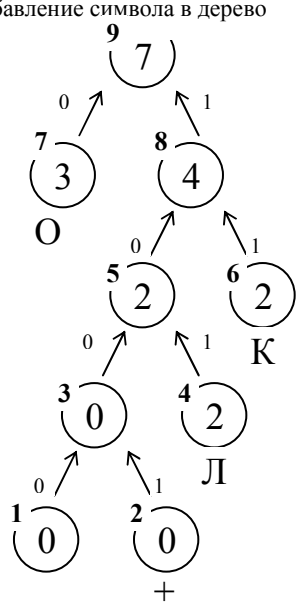
Упорядочивание дерева



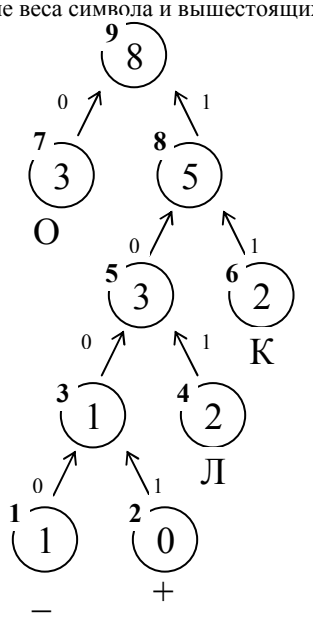
Итерация 8. Добавление символа "_"

Проверяем наличие символа "_" в текущем дереве и формируем код 100 1000 (код символа "+" в текущем дереве, полученном после итерации 7 и исходный код символа "_")

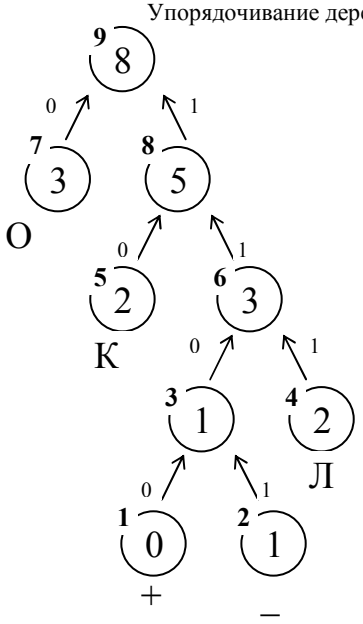
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов



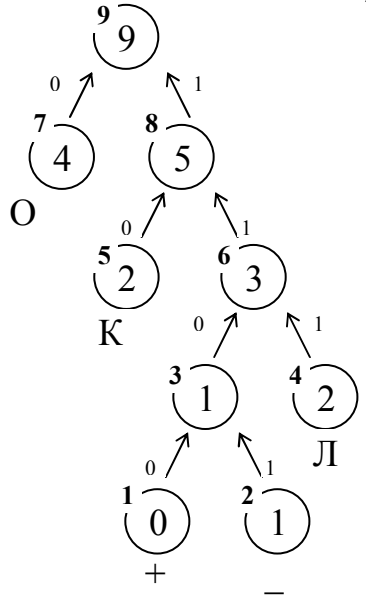
Упорядочивание дерева



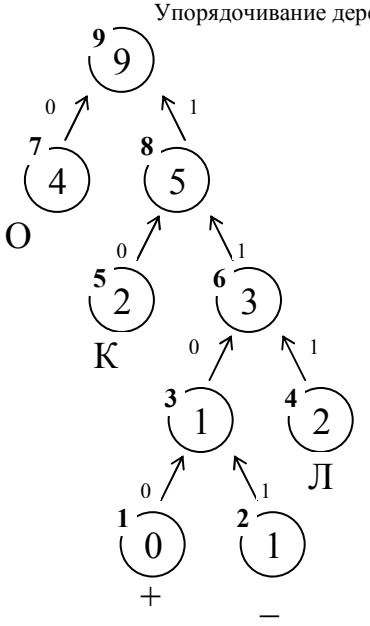
Итерация 9. Добавление символа "О"

Проверяем наличие символа "О" в дереве и формируем код 0 (код символа "О" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



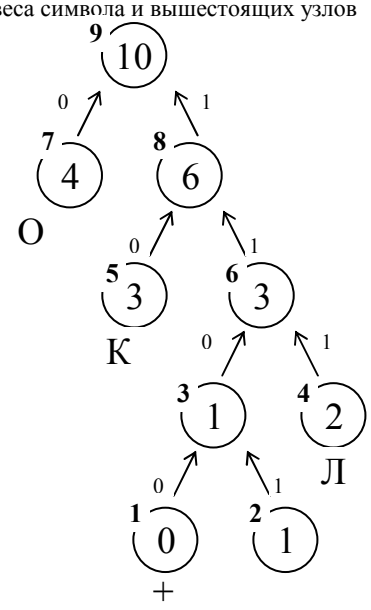
Упорядочивание дерева



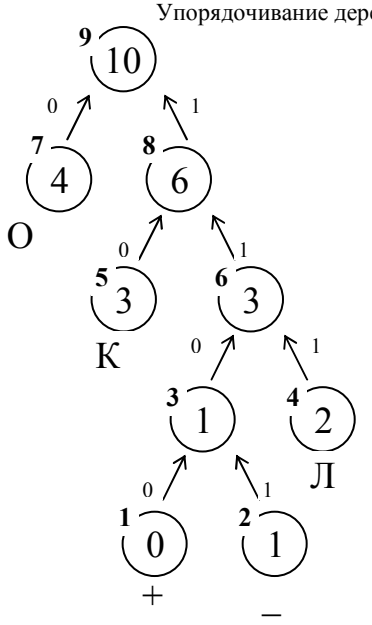
Итерация 10. Добавление символа "К"

Проверяем наличие символа "К" в дереве и формируем код 10 (код символа "К" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



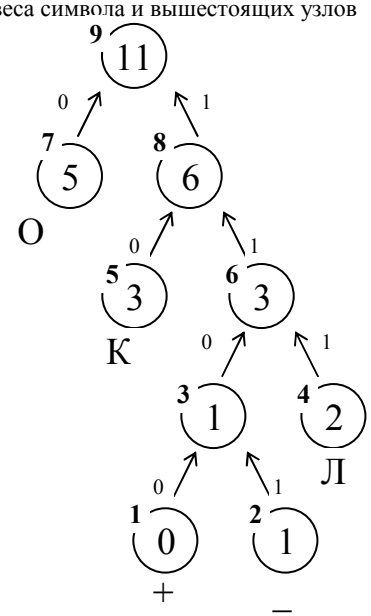
Упорядочивание дерева



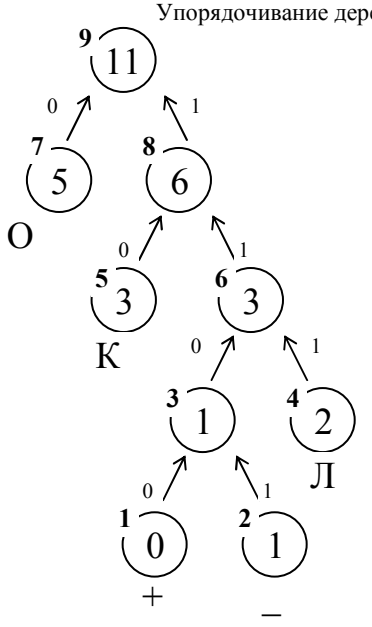
Итерация 11. Добавление символа "О"

Проверяем наличие символа "О" в дереве и формируем код 0 (код символа "О" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



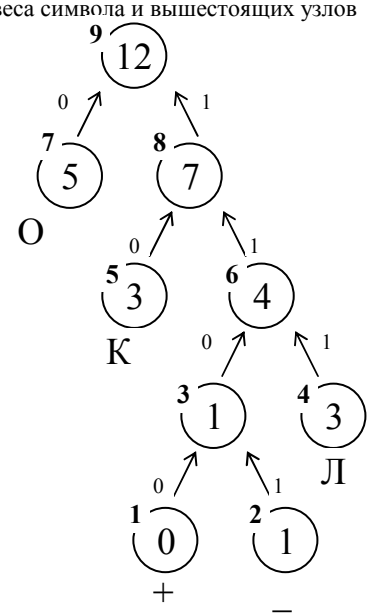
Упорядочивание дерева



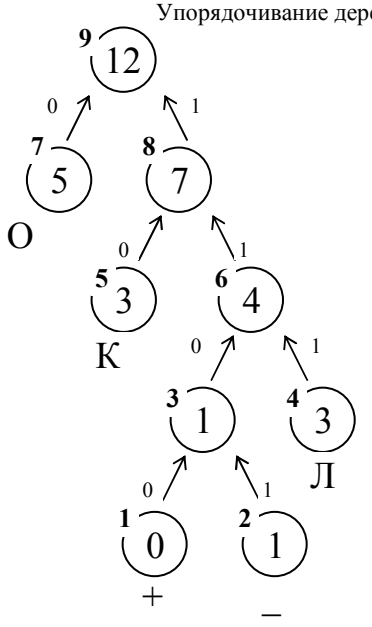
Итерация 12. Добавление символа "Л"

Проверяем наличие символа "Л" в дереве и формируем код 111 (код символа "Л" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



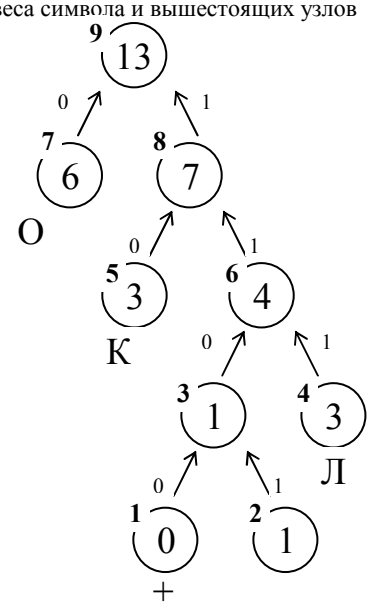
Упорядочивание дерева



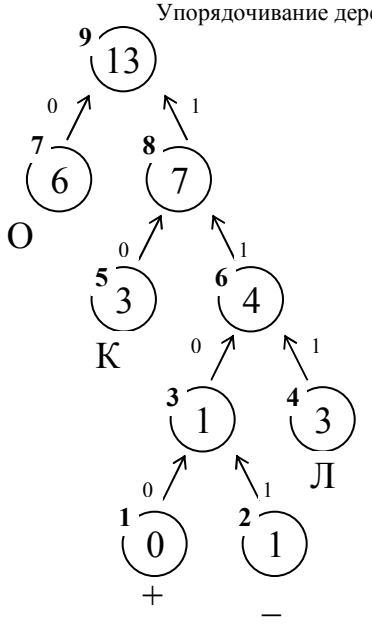
Итерация 13. Добавление символа "О"

Проверяем наличие символа "О" в дереве и формируем код 0 (код символа "О" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



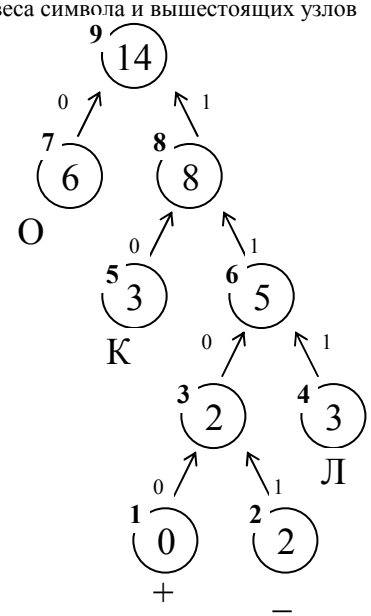
Упорядочивание дерева



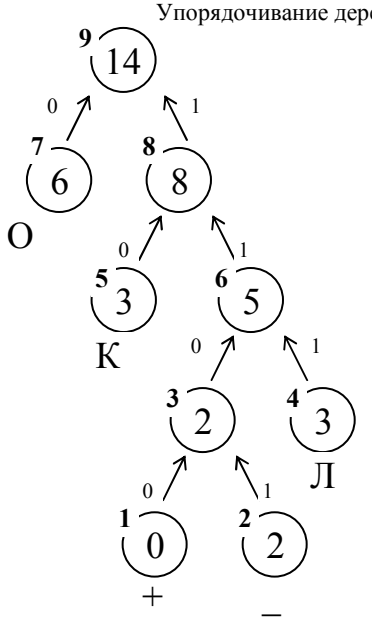
Итерация 14. Добавление символа "_"

Проверяем наличие символа "_" в дереве и формируем код 1101 (код символа "_" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



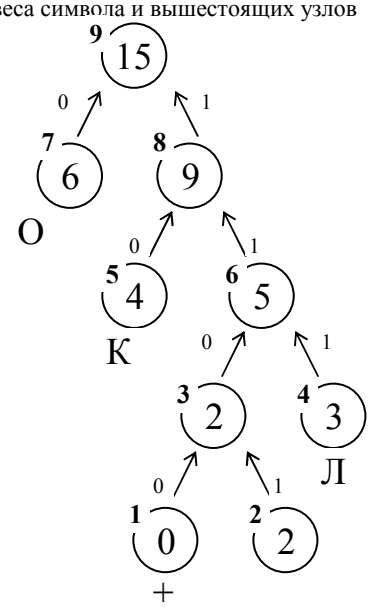
Упорядочивание дерева



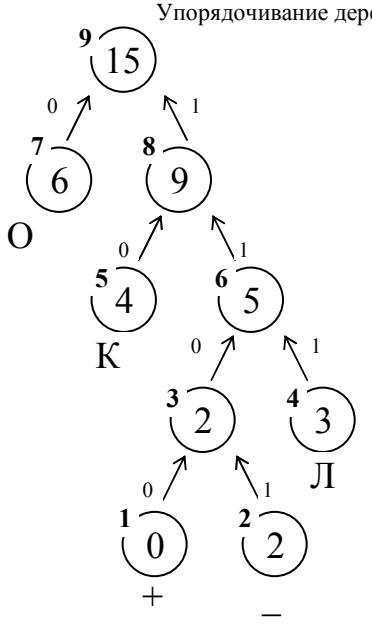
Итерация 15. Добавление символа "К"

Проверяем наличие символа "К" в дереве и формируем код 10 (код символа "К" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



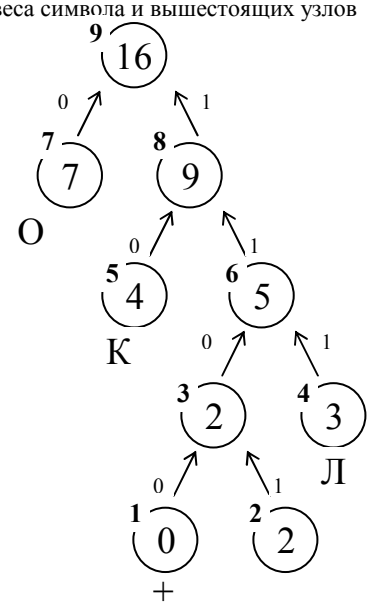
Упорядочивание дерева



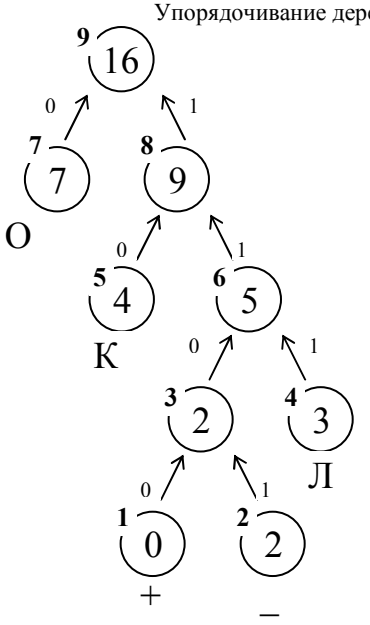
Итерация 16. Добавление символа "О"

Проверяем наличие символа "О" в дереве и формируем код 0 (код символа "О" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



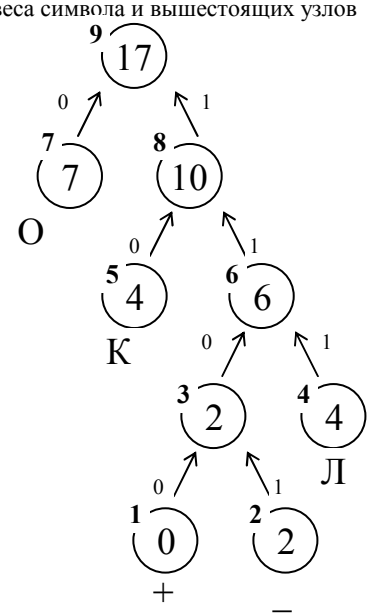
Упорядочивание дерева



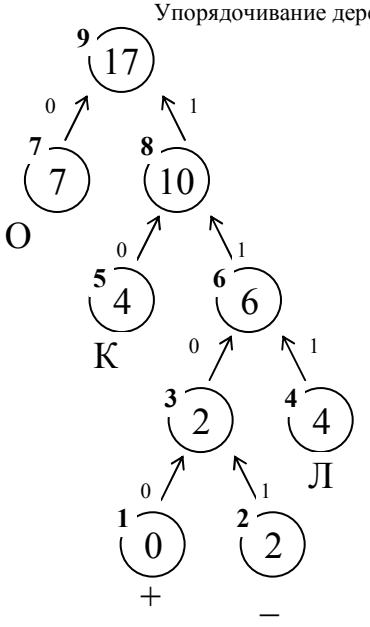
Итерация 17. Добавление символа "Л"

Проверяем наличие символа "Л" в дереве и формируем код 111 (код символа "Л" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



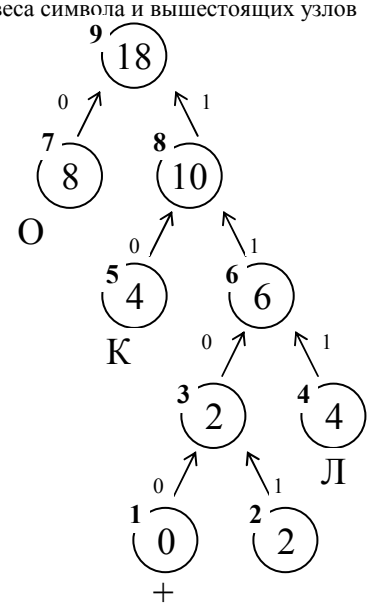
Упорядочивание дерева



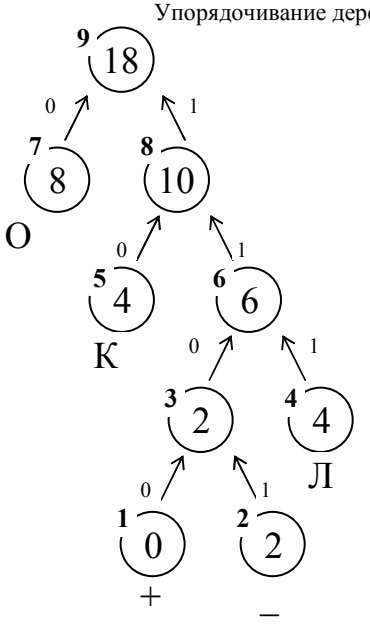
Итерация 18. Добавление символа "О"

Проверяем наличие символа "О" в дереве и формируем код 0 (код символа "О" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



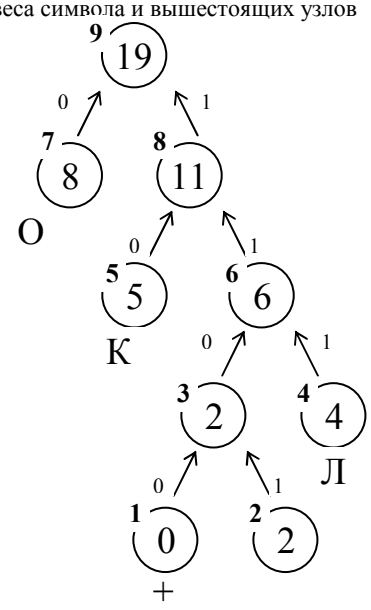
Упорядочивание дерева



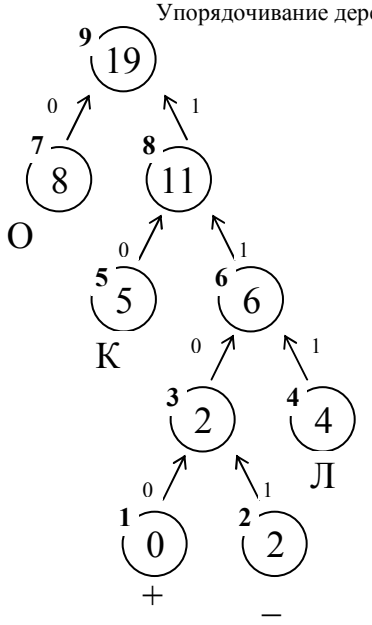
Итерация 19. Добавление символа "К"

Проверяем наличие символа "К" в дереве и формируем код 10 (код символа "К" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



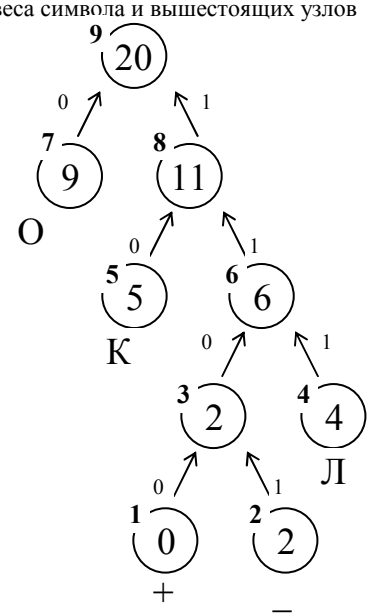
Упорядочивание дерева



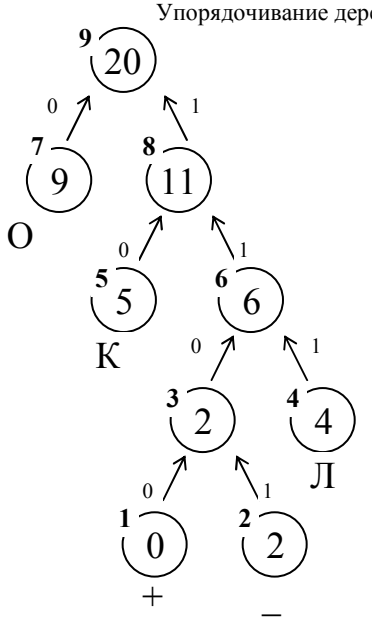
Итерация 20. Добавление символа "О"

Проверяем наличие символа "О" в дереве и формируем код 0 (код символа "О" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



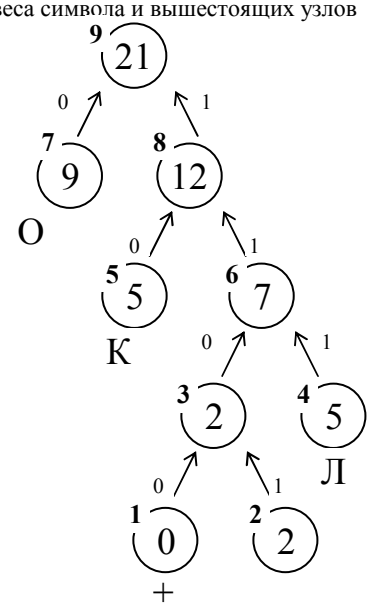
Упорядочивание дерева



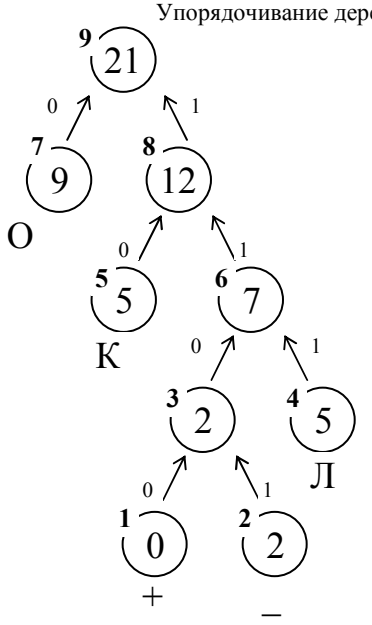
Итерация 21. Добавление символа "Л"

Проверяем наличие символа "Л" в дереве и формируем код 111 (код символа "Л" в текущем дереве)

Увеличение веса символа и вышестоящих узлов



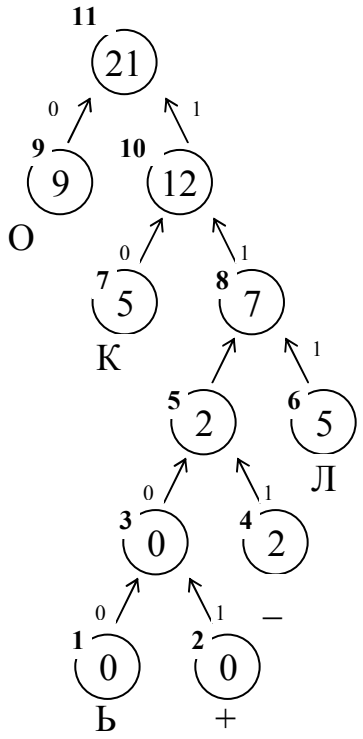
Упорядочивание дерева



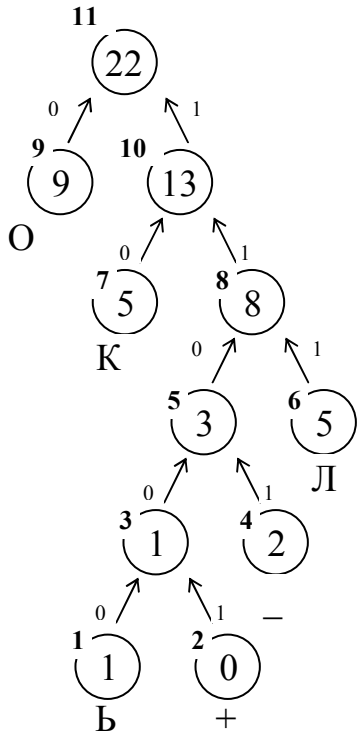
Итерация 22. Добавление символа "Ь"

Проверяем наличие символа "Ь" в текущем дереве и формируем код 1100 0101 (код символа "+" в текущем дереве, полученном после итерации 21 и исходный код символа "Ь")

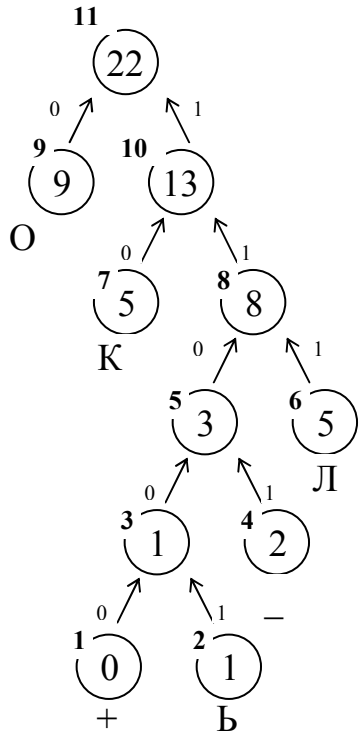
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов



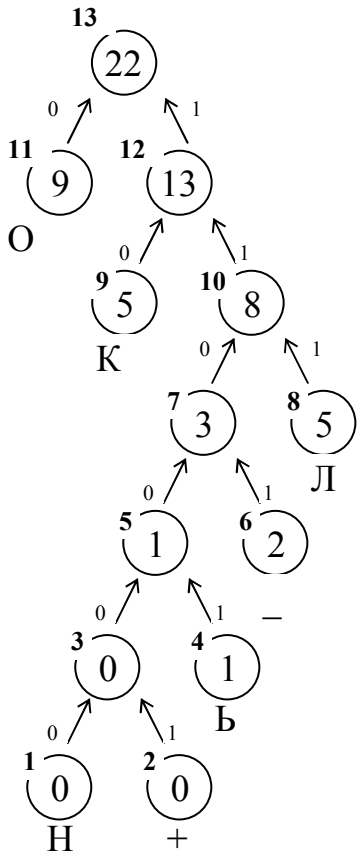
Упорядочивание дерева



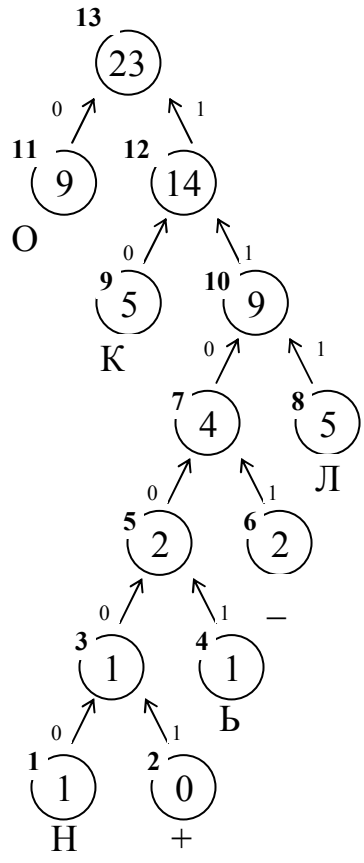
Итерация 23. Добавление символа "Н"

Проверяем наличие символа "Н" в текущем дереве и формируем код 11000 0011 (код символа "+" в текущем дереве, полученном после итерации 22 и исходный код символа "Н")

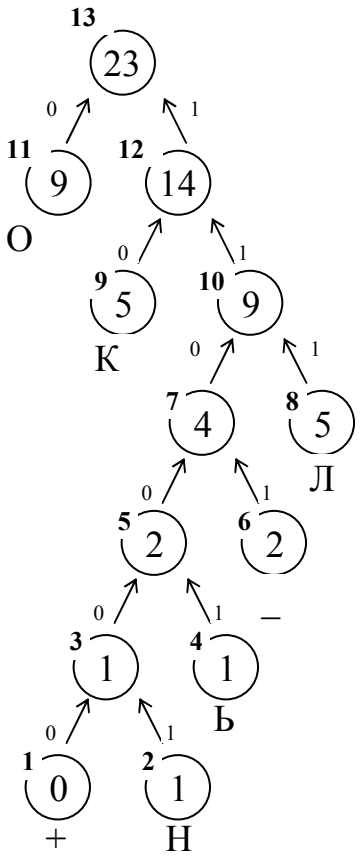
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов

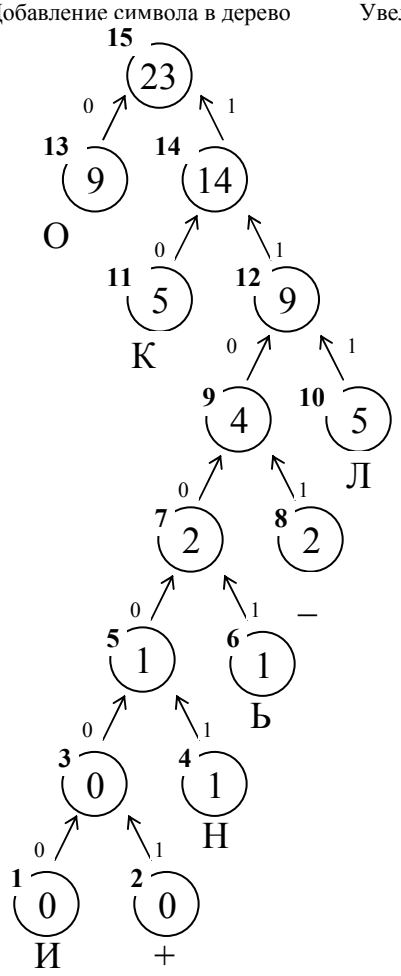


Упорядочивание дерева

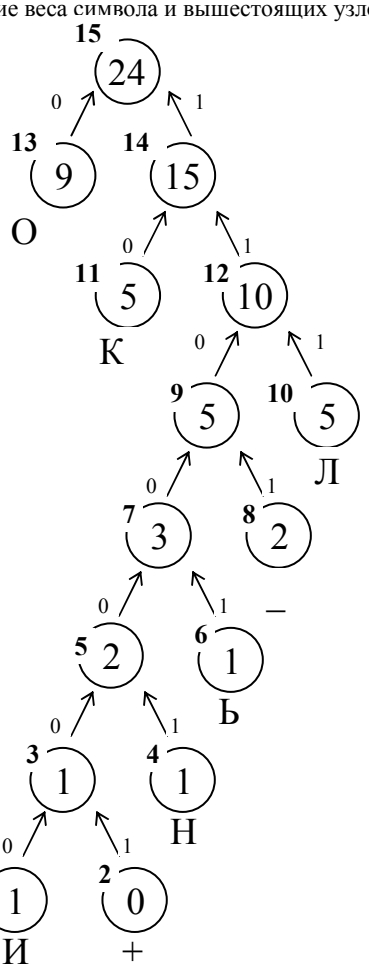


Итерация 24. Добавление символа "И". Проверяем наличие символа "И" в текущем дереве и формируем код 110000 0000

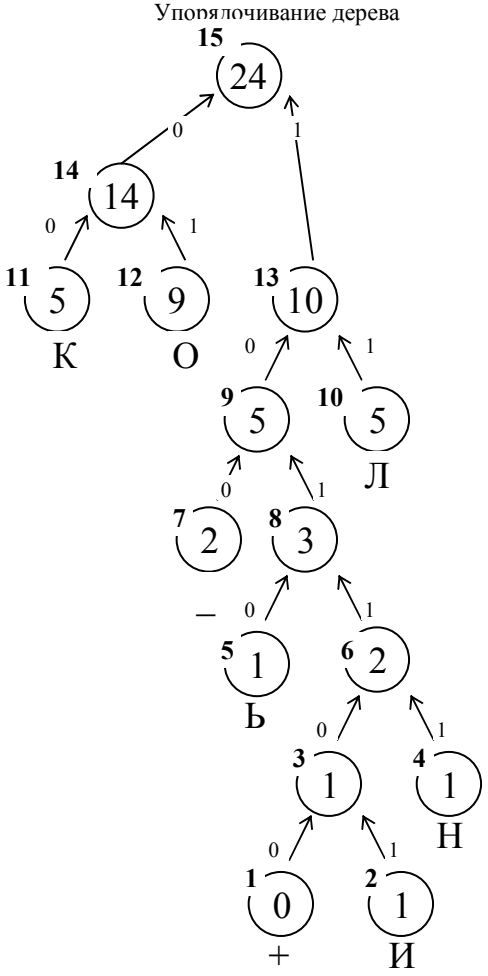
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов

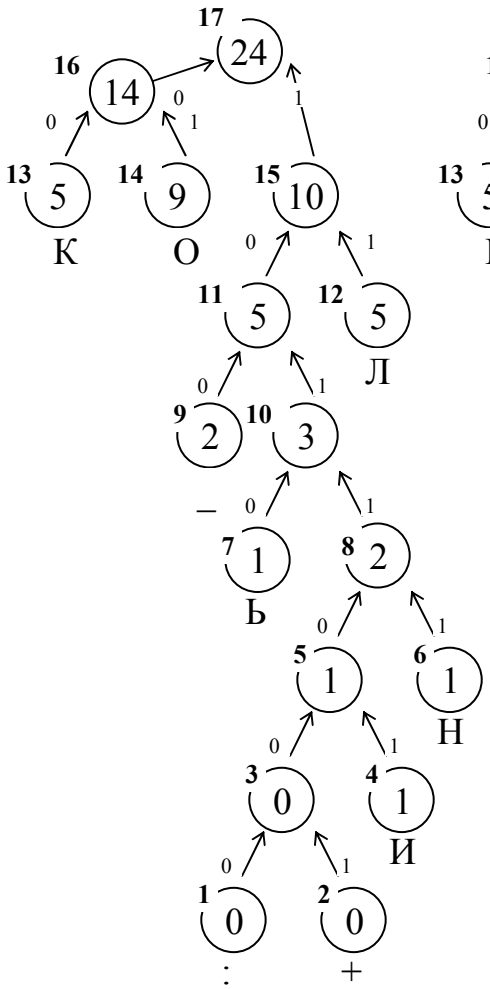


Упорядочивание дерева

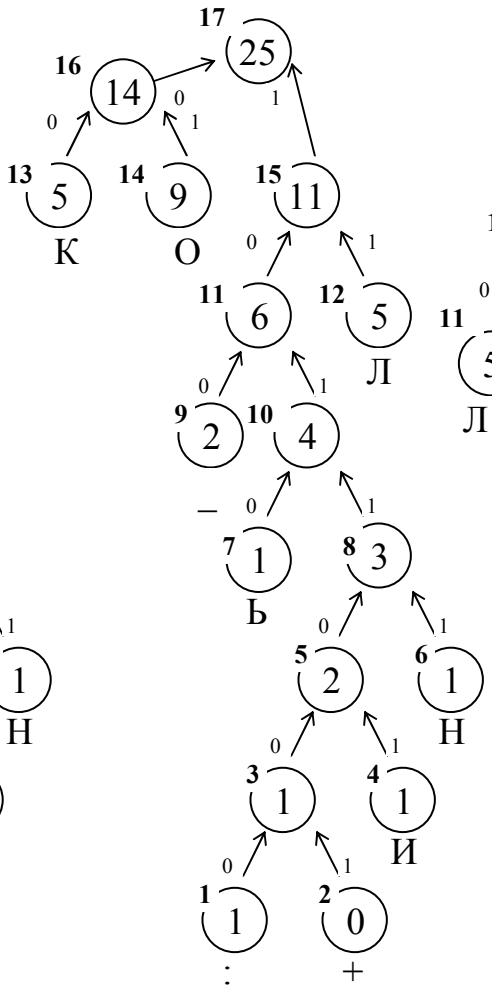


Итерация 25. Добавление символа ":". Проверяем наличие символа ":" в текущем дереве и формируем код 101100 0110

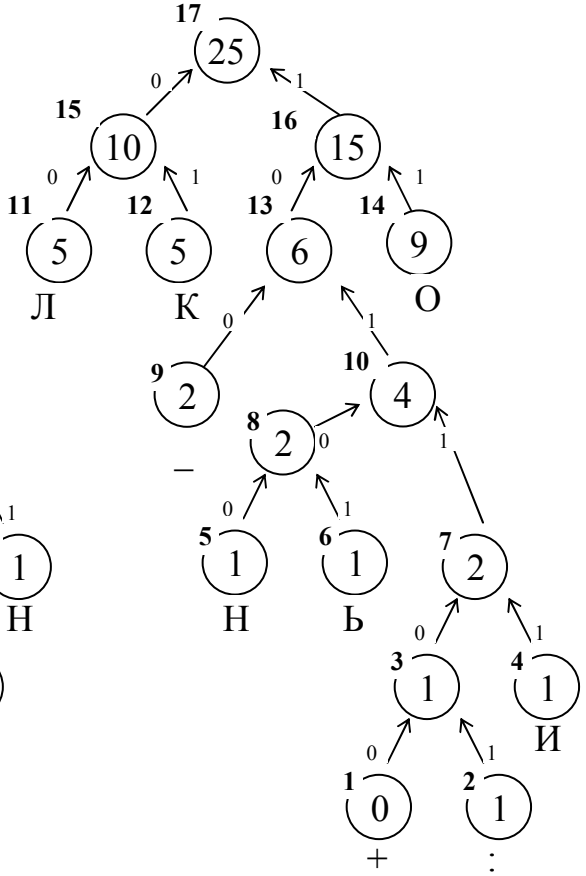
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов

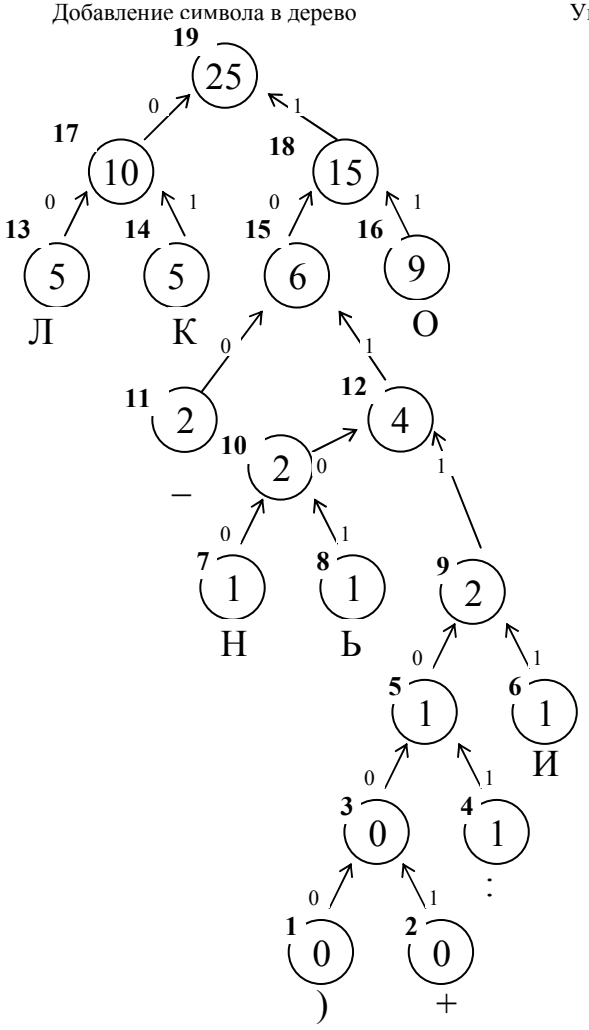


Упорядочивание дерева

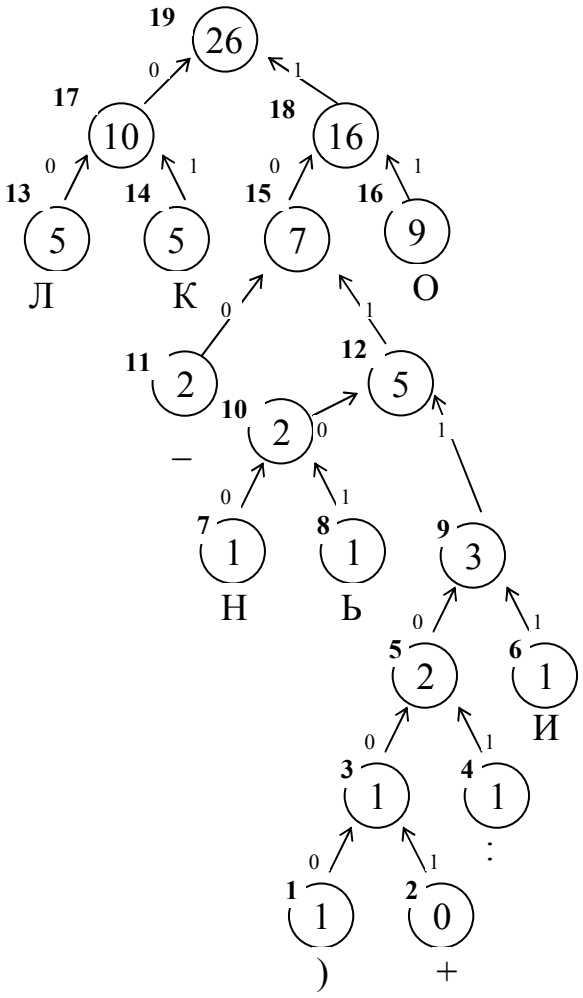


Итерация 26. Добавление символа ")". Проверяем наличие символа ")" в текущем дереве и формируем код 101100 0111

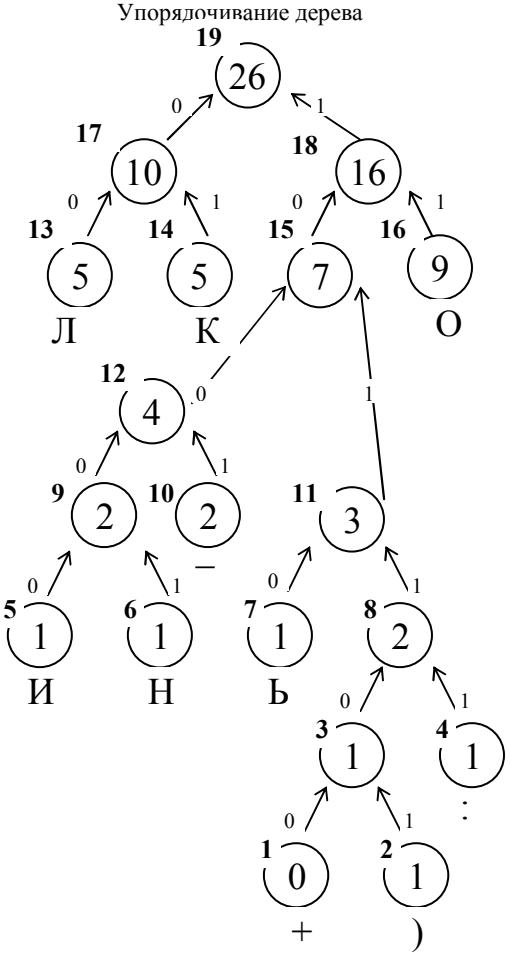
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов



Упорядочивание дерева



Код Хаффмана, полученный в результате адаптивного кодирования

К	О	Л	О	К	О	Л	_	О	К	О	Л	О	_
0001	0 0100	00 0010	11	11	0	101	100 1000	0	10	0	111	0	1101

К	О	Л	О	К	О	Л	Ь	Н	И	:	)
10	0	111	0	10	0	111	1100 0101	11000 0011	110000 0000	101100 0110	101100 0111

Объем исходного (несжатого) текста в равномерной кодировке $V_{исх}$ = 104 бита
Объем сжатого текста $V_{сж}$ = 102 бита

Распаковка адаптивного кода Хаффмана и восстановление исходного текста

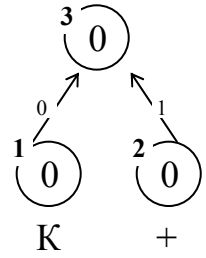
Исходные коды постоянной длины m (4 бита)

И	0	0	0	0	**Н**	0	0	1	1	**:**	0	1	1	0
К	0	0	0	1	**О**	0	1	0	0	**)**	0	1	1	1
Л	0	0	1	0	**Ь**	0	1	0	1	**_**	1	0	0	0

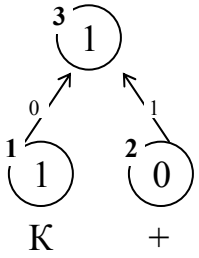
Итерация 0. Создание пустого дерева.

Итерация 1. Декодирование первого символа. Из битого потока 0001 0 0100… выделяем первые 4 бита, определяем по таблице исходных кодов символ "К", добавляем его в пустое дерево

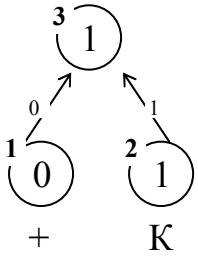
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов

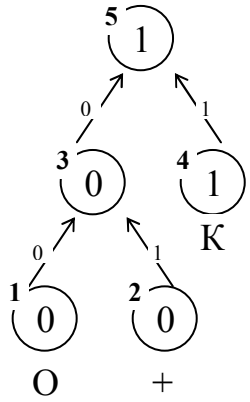


Упорядочивание дерева

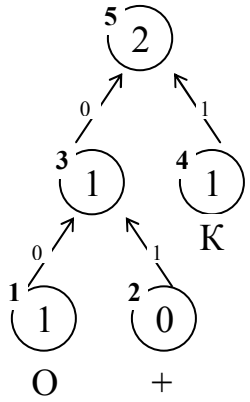


Итерация 2. Декодирование второго символа. По первому биту "0" в текущем битовом потоке …0 0100 00 0010… определяем, что был закодирован новый символ. Выделяем 4 бита, определяем по таблице исходных кодов символ "О", добавляем его в дерево, полученное после итерации 1.

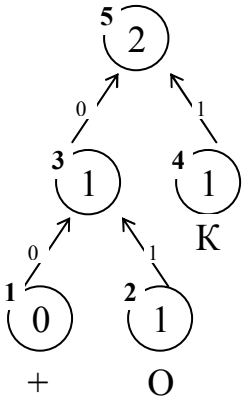
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов

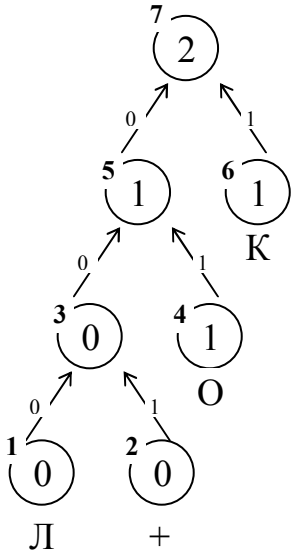


Упорядочивание дерева

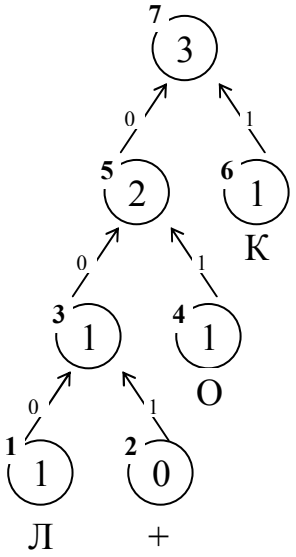


Итерация 3. Декодирование третьего символа. По коду "00" в текущем битовом потоке …00 0010 11… определяем, что был закодирован новый символ. Выделяем 4 бита, определяем по таблице исходных кодов символ "Л", добавляем его в дерево, полученное после итерации 2.

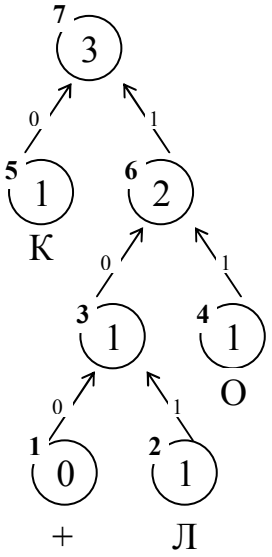
Добавление символа в дерево



Увеличение веса символа и вышестоящих узлов

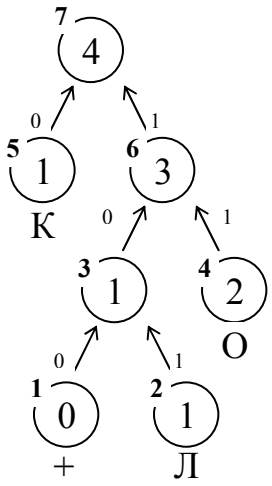


Упорядочивание дерева



Итерация 4. Декодирование четвертого символа. По коду "11" в текущем битовом потоке …11 11 0… определяем, что был закодирован символ "О", уже содержащийся в дереве.

Увеличение веса символа и вышестоящих узлов



Упорядочивание дерева