6. Адаптивное кодирование

6.1 Идея адаптивного кодирования

Методы сжатия данных без потерь можно разделить на два основных типа:

- **статическое (блочное) сжатие**
- **динамическое (адаптивное или потоковое) сжатие**

Основное отличие статических и динамических алгоритмов состоит в следующем:

Статические алгоритмы обрабатывают данные порциями (блоками), причем обработка блока требует, как минимум, двух проходов. На первом проходе данные анализируются, а на втором преобразуются в упакованный вид.

- **Динамические алгоритмы обрабатывают поток данных за один проход.**

Второе отличие:

- **Статический алгоритм требует записи специальной вспомогательной информации для каждого упакованного блока данных и упаковка/распаковка осуществляется поблочно (порциями).**
- **Динамический алгоритм не требует записи специальной вспомогательной информации для распаковки и распаковка также осуществляется потоком.**

- **В результате статические методы неприменимы там, где поток данных невозможно прервать для обработки порции данных.**
- **Кроме того требуется располагать достаточно большим буфером на обоих концах линии для хранения блока данных.**

Практически любой статический метод можно преобразовать в динамический по следующему алгоритму:

1. **Вспомогательная информация для кодирования (упаковки/распаковки) инициализируется наперед заданным образом.**
2. **Очередной байт данных кодируется в соответствии с текущим состоянием вспомогательной информации.**
3. **Вспомогательная информация для кодирования изменяется в соответствии с обработанным байтом.**
4. **Процесс повторяется с шага 2., пока данные не закончились.**

6.2 Приемы для оценки статистики

- **Для каждого символа *x*, порождаемого источником, количество появлений этого символа хранится в счетчике C(x).**
- **Перед кодированием C(x)=1 (или 0) для всех x.**
- **При появлении символа x в сообщении C(x)= C(x)+1.**

- **Кодирование очередного символа сообщения происходит с учетом текущих значений счетчиков.**

- **При кодировании достаточно больших сообщений может происходить переполнение счетчиков.**
- **Для исправления этой ситуации**
  - **счетчики сбрасываются**
  - **счетчики масштабируются**

- **Большинство адаптивных методов для учета изменений статистики исходных данных используют так называемое *окно.***
- ***Окном* называют последовательность символов, предшествующих кодируемой букве, а *длиной окна* – количество символов в окне.**

- **Обычно окно имеет фиксированную длину и после кодирования каждой буквы сообщения окно передвигается на один символ вправо.**
- **Таким образом, код для очередной буквы строится с учетом информации, хранящейся в данный момент в окне.**


кодируемый символ
... a3 a2 a5 a2 a4 a1 a1 a3 a2 a2 a2 a4 a3 a1 a2 ...
кодируемый символ
... a3 a2 a5 a2 a4 a1 a1 a3 a2 a2 a2 a4 a3 a1 a2 ...
кодируемый символ
... a3 a2 a5 a2 a4 a1 a1 a3 a2 a2 a2 a4 a3 a1 a2 ...

- **При декодировании окно передвигается по закодированному сообщению аналогичным образом.**
- **Информация, содержащаяся в окне, позволяет однозначно декодировать очередной символ.**

Оценка избыточности при адаптивном кодировании складывается из двух составляющих:

- **избыточность кодирования и**
- **избыточность, возникающая при оценке вероятностей появления символов,**

Эффективность методов адаптивного кодирования зачастую оценивают экспериментальным путем.

Для методов адаптивного кодирования справедлива следующая теорема:

Теорема. *Величина средней длины кодового слова при адаптивном кодировании удовлетворяет неравенству*

$$L_{ср} \leq H + C$$

где – H энтропия источника информации, C – константа, зависящая от размера алфавита источника и длины окна.

6.3 Динамический код Хаффмана

Классический алгоритм Хаффмана имеет один существенный недостаток

- **Для восстановления содержимого сообщения декодер должен знать таблицу частот, которой пользовался кодер.**
- **Следовательно, длина сжатого сообщения увеличивается на длину таблицы частот, которая должна посылаться впереди данных, что может свести все усилия по сжатию сообщения.**

- **Кроме того необходимость наличия полной частотной статистики перед началом собственно кодирования требует двух проходов по сообщению:**

  для построения модели сообщения

  для кодирования сообщения.

- **Для оценки статистики источника будут использоваться счетчики появлений символов.**
- **Поскольку при поступлении следующего символов счетчики изменяются незначительно, то нет необходимости полностью строить дерево кодирования Хаффмана.**
- **Достаточно перестроить уже имеющееся.**

- **Дерево кодирования обладает свойством упорядоченности, если его узлы могут быть перечислены в порядке возрастания веса и в этом перечислении каждый узел находится рядом со своим братом.**
- **Двоичное дерево является деревом кодирования Хаффмана тогда и только тогда, когда оно удовлетворяет свойству упорядоченности.**

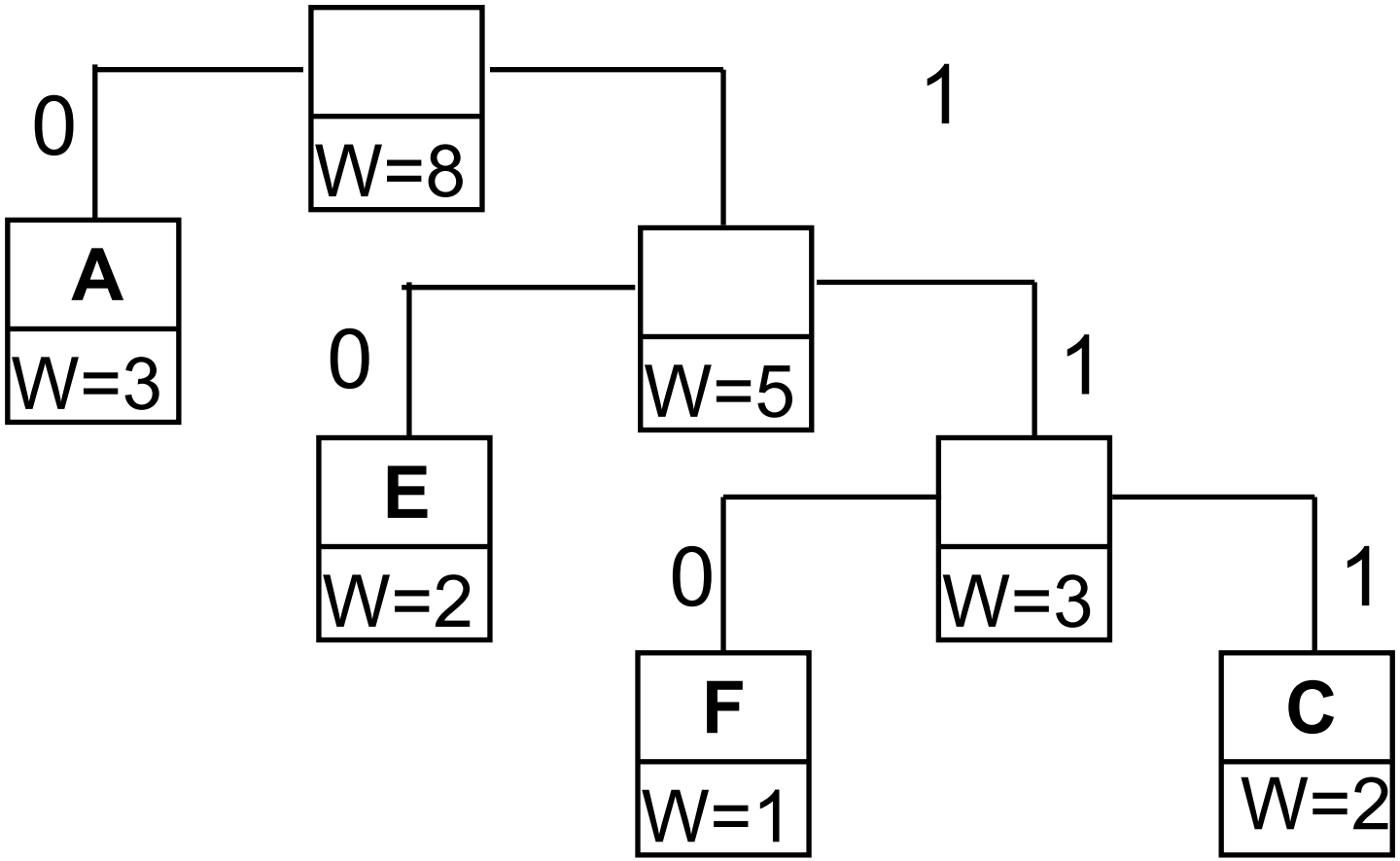
- **Закодируем сообщение**

  A A F A C E E C

  Вместо частот (вероятностей) символов используются счетчики вхождений символов в сообщение

Дерево кодирования

A A F A C E E C



Обновление дерева при считывании очередного символа сообщения состоит из двух операций

Увеличение веса узлов дерева.

- **Вначале увеличиваем вес листа, соответствующего считанному символу, на единицу.**
- **Затем увеличиваем вес родителя, чтобы привести его в соответствие с новыми значениями веса у детей. Этот процесс продолжается до тех пор, пока не доберемся до корня дерева.**
- **Среднее число операций увеличения веса равно среднему количеству битов, необходимых для того, чтобы закодировать символ.**

Перестановка узлов дерева требуется тогда, когда увеличение веса узла приводит к нарушению свойства упорядоченности, то есть тогда, когда увеличенный вес узла стал больше, чем вес следующего по порядку узла.

Если и дальше продолжать обрабатывать увеличение веса, двигаясь к корню дерева, то дерево перестанет быть деревом Хаффмана.

Чтобы сохранить упорядоченность дерева кодирования, алгоритм работает следующим образом.

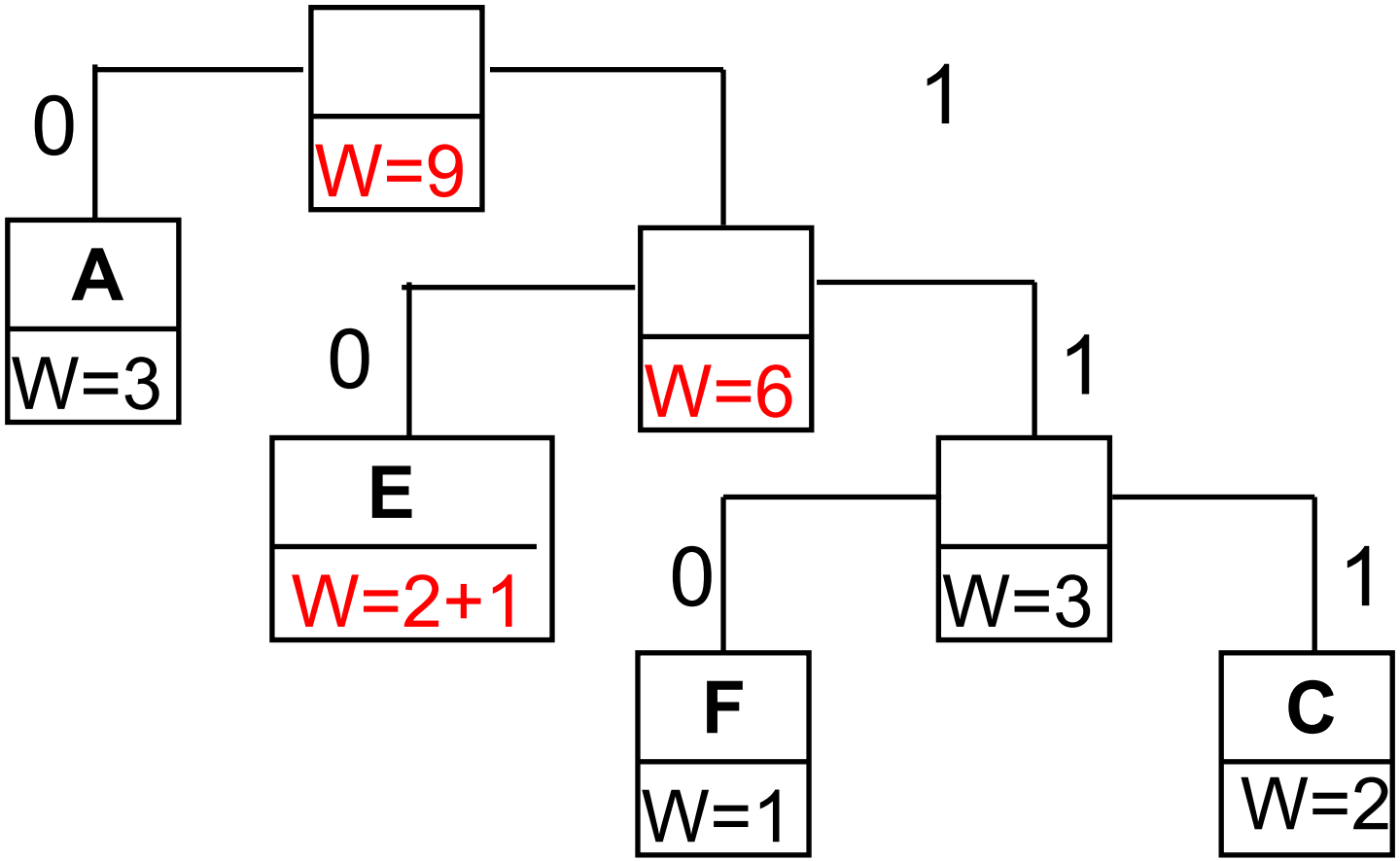
- **Пусть новый увеличенный вес узла равен W+1. Тогда начинаем двигаться по списку в сторону увеличения веса, пока не найдем последний узел с весом W.**
- **Переставим текущий и найденный узлы между собой в списке, восстанавливая таким образом порядок в дереве.**

- **Добавим символ E к сообщению**

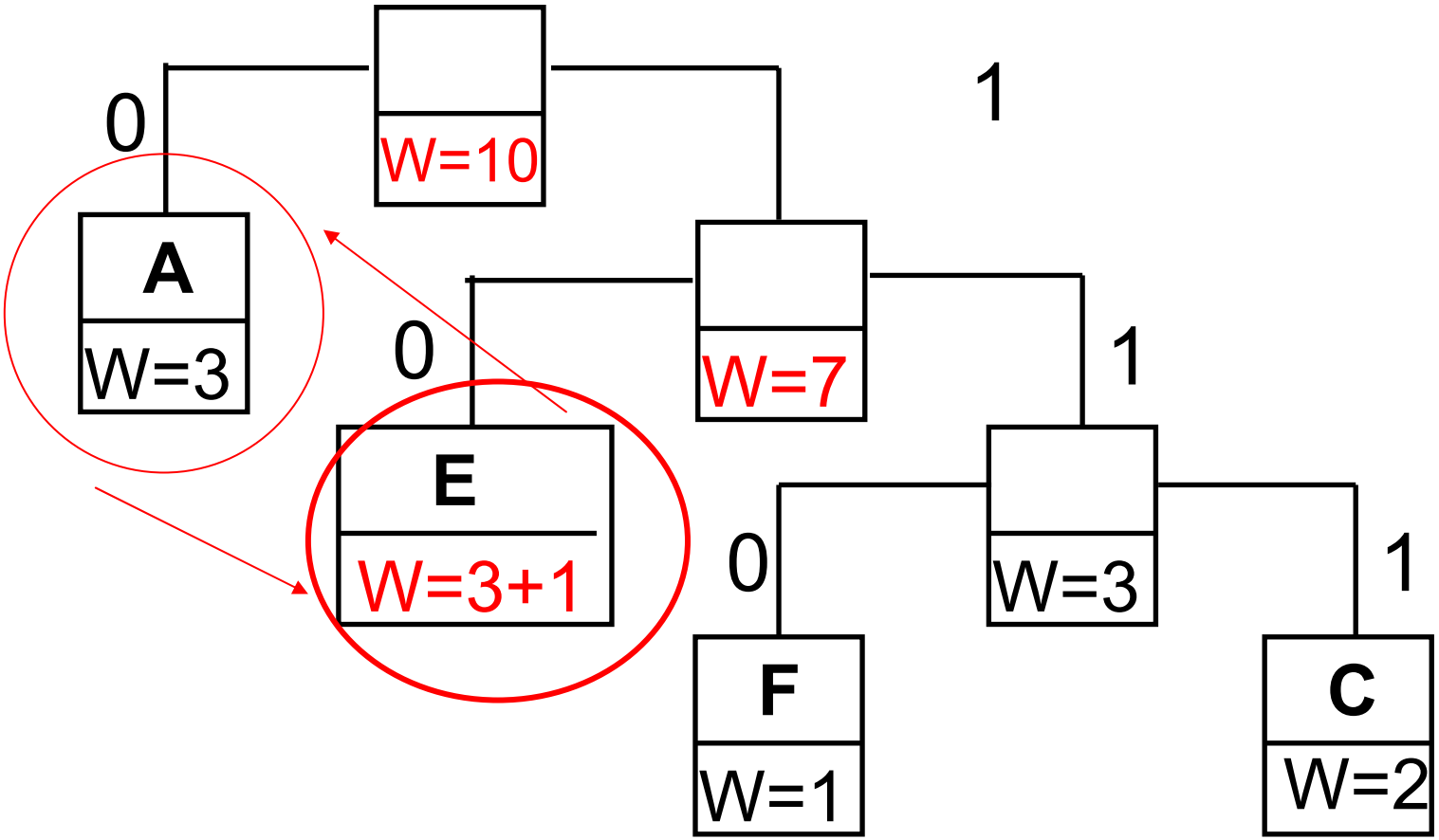
A A F A C E E C E

Тогда дерево изменится следующим образом

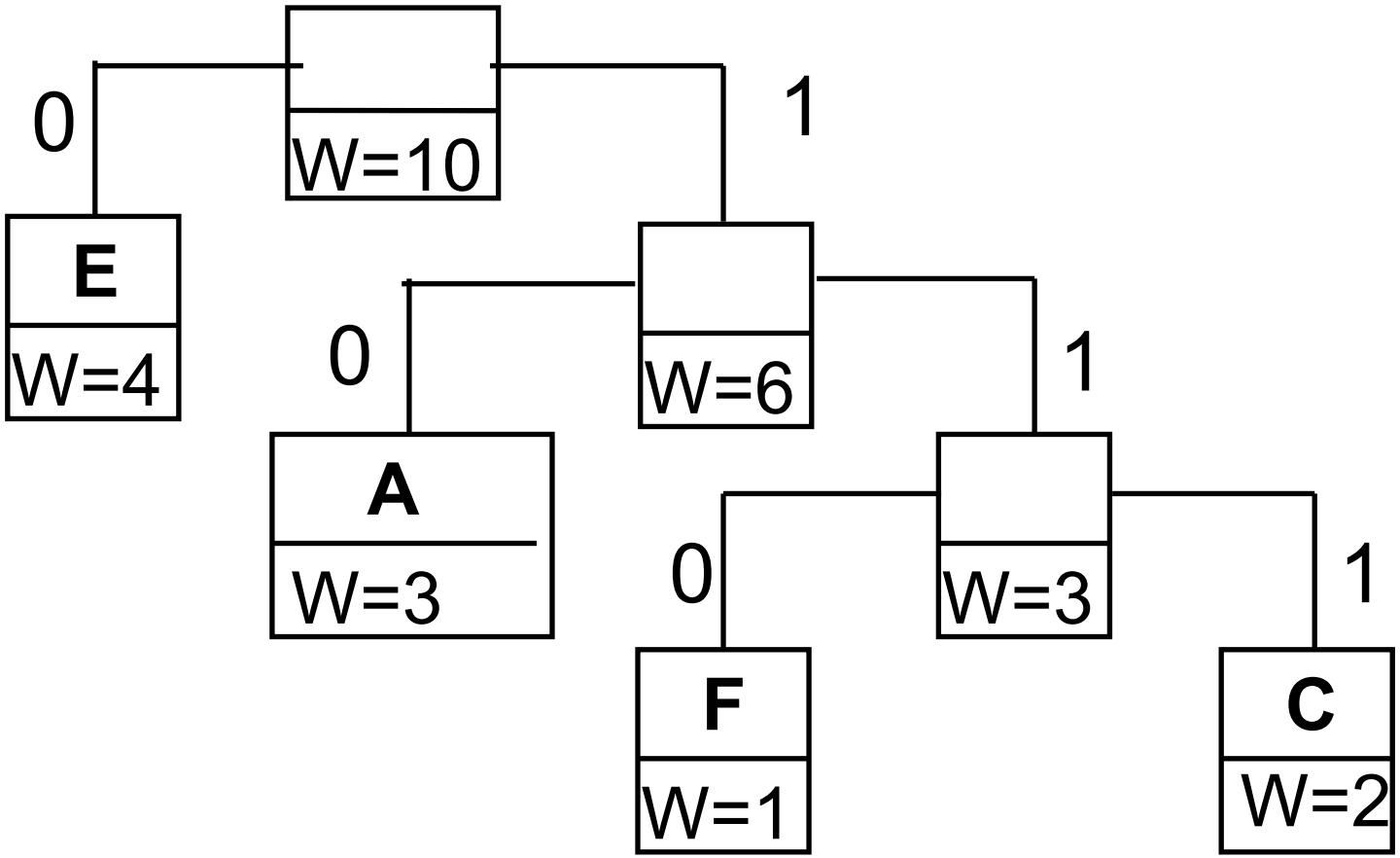
Добавлен символ E



Добавлен символ E



Меняем местами вершины A и E



- **В классическом алгоритме Хаффмана, те символы, которые не встречаются в сообщении, уже известны до начала кодирования, поскольку известна таблица частот.**
- **В адаптивной версии алгоритма заранее неизвестно какие символы появятся в сообщении.**

- **Можно проинициализировать дерево Хаффмана так, чтобы оно имело все 256 символов алфавита (для 8-и битовых кодов) с частотой, равной 1 (или 0).**

- **В начале кодирования каждый код будет иметь длину 8 бит.**
- **По мере адаптации модели наиболее часто встречающиеся символы будут кодироваться все меньшим и меньшим количеством битов.**
- **Такой подход работоспособен, но он значительно снижает степень сжатия, особенно на коротких сообщениях.**

- **Лучше начинать моделирование с пустого дерева и добавлять в него символы только по мере их появления в сжимаемом сообщении.**
- **Но когда символ появляется в сообщении первый раз, он не может быть закодирован, так как его еще нет в дереве кодирования.**

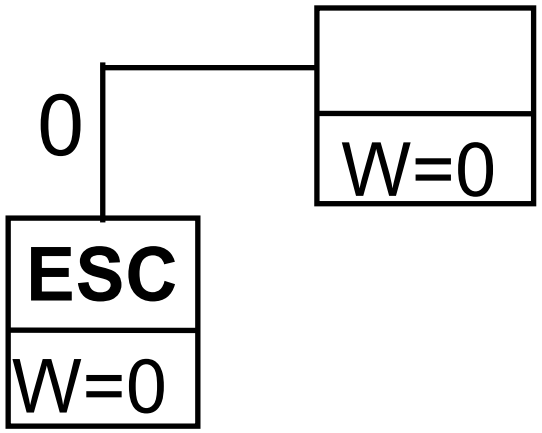
- **Чтобы разрешить это противоречие, вводится специальный ESC-код, который будет означать, что следующий символ закодирован вне контекста модели сообщения.**
- **Например, его можно передать в поток сжатой информации как есть, не кодируя вообще.**

- **Использование специального символа ESC подразумевает определенную инициализацию дерева до начала кодирования и декодирования: в него помещаются 2 специальных символа: ESC и EOF (конец файла), с весом, равным 1 (или 0)**
- **Поскольку процесс обновления не коснется их веса, то по ходу кодирования они будут перемещаться на самые удаленные ветви дерева и иметь самые длинные коды.**

Метод "ЗакодироватьСимвол" в алгоритме адаптивного кодирования Хаффмана можно записать следующим образом.

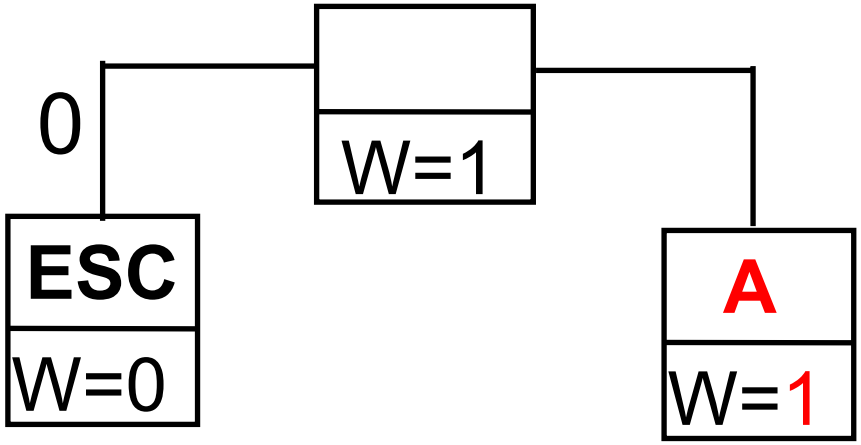
- ***ЗакодироватьСимвол(Символ)***
- ***{ Если СимволУжеЕстьВТаблице(Символ) ВыдатьКодХаффманаДля(Символ); Иначе***
- ***{ ВыдатьКодХаффманаДля(ESC); ВыдатьСимвол(Символ);***
- ***}***
- ***}***

Начальное дерево кодирования и декодирования

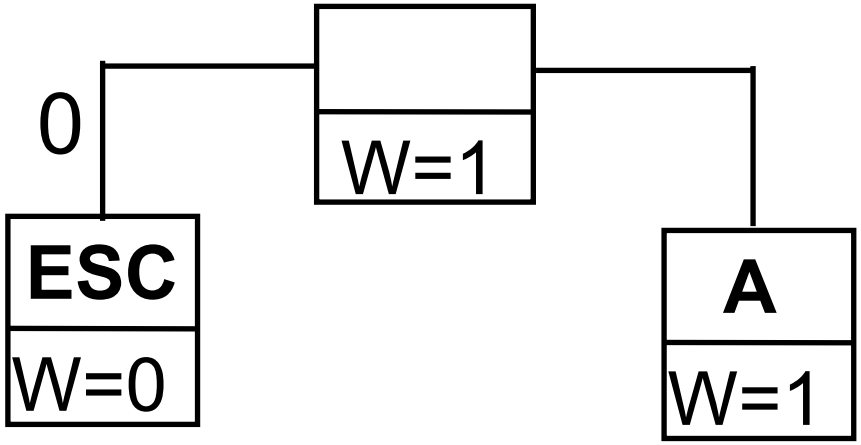


AAFACEECC

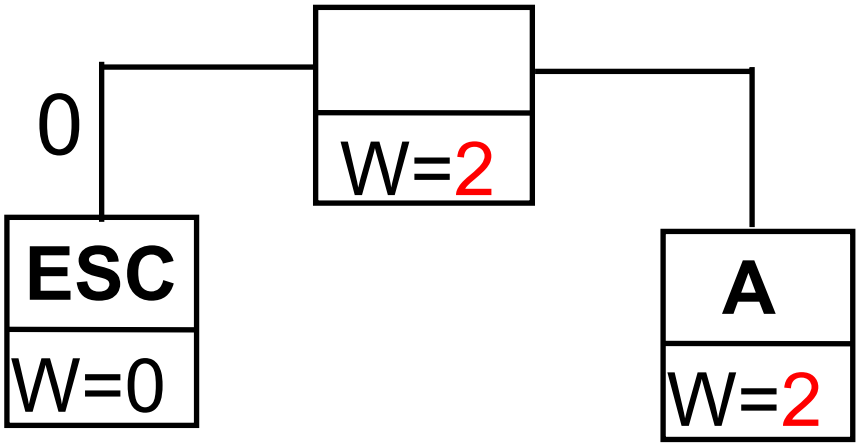
Для A передается код ESC‘A’


0
W=1
ESC
W=0
A
W=1

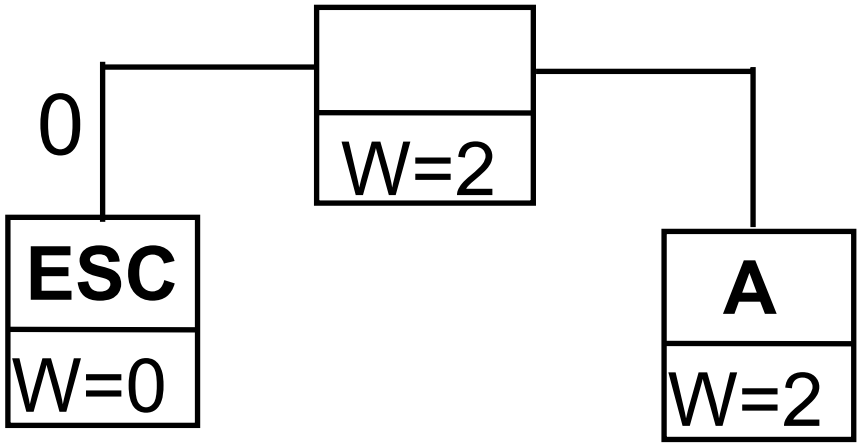
AAFACEECC



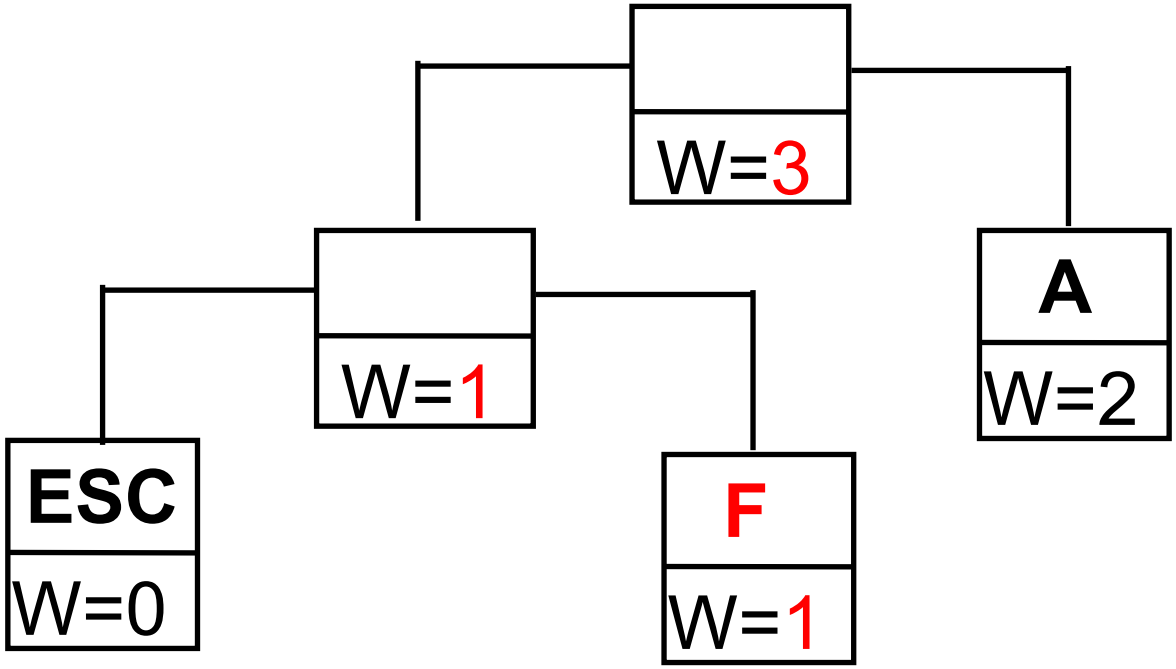
Для A передается код 1


0
W=2
ESC
W=0
A
W=2

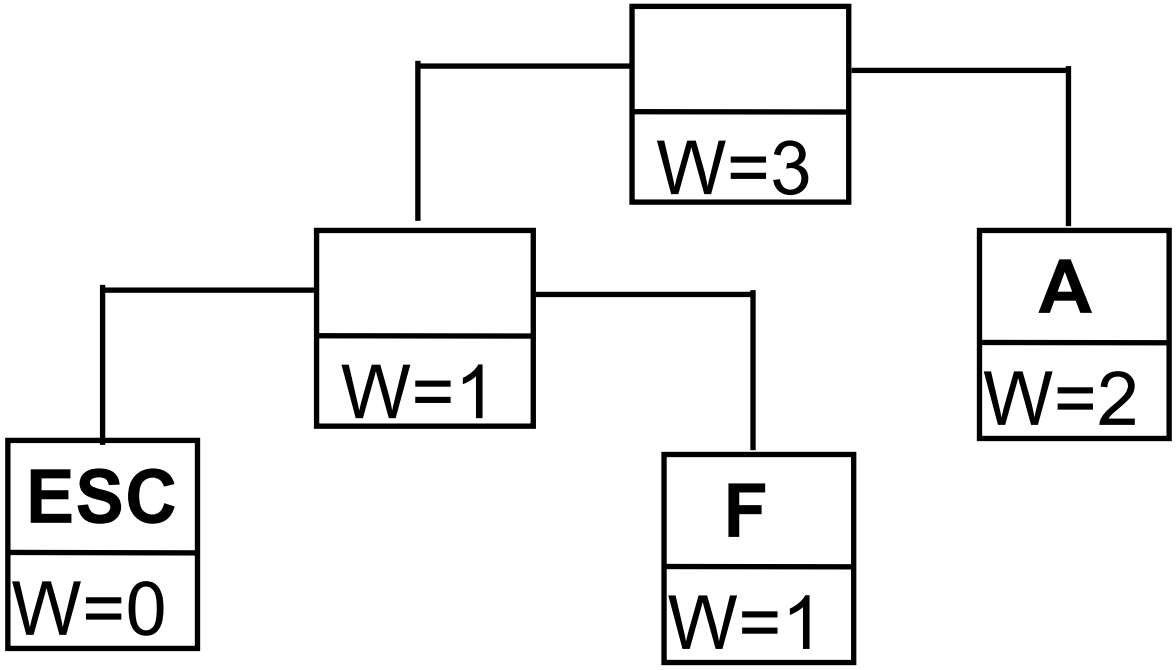
AAFACEECC



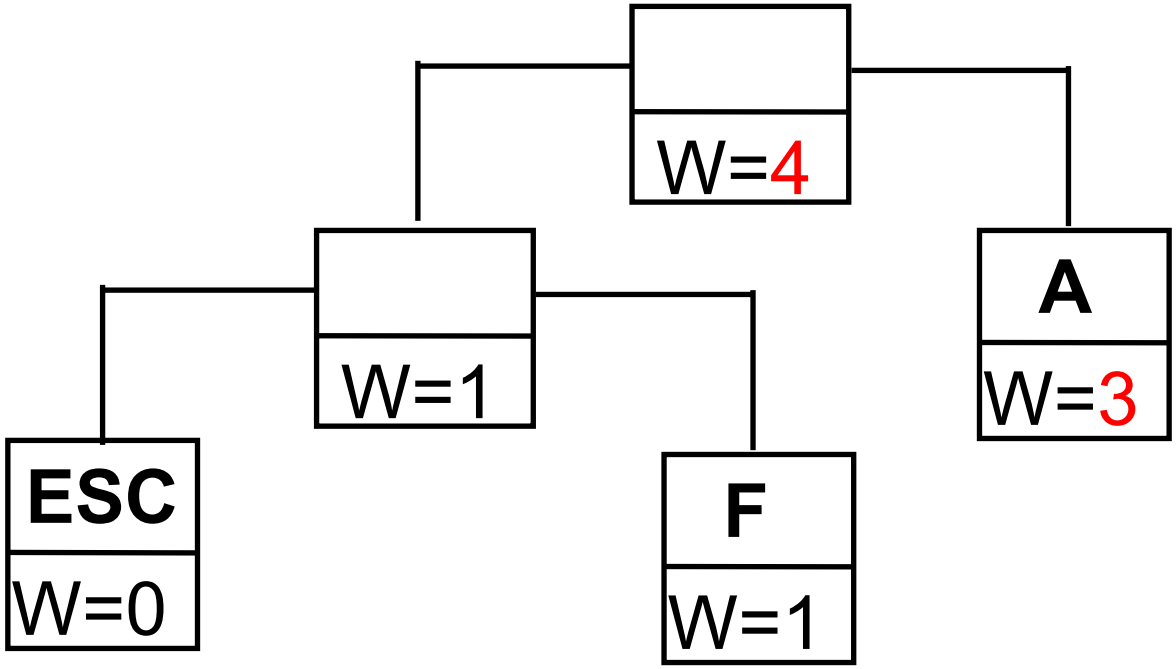
Для F передается код 0’F’


W=3
W=1
A
W=2
ESC
W=0
F
W=1

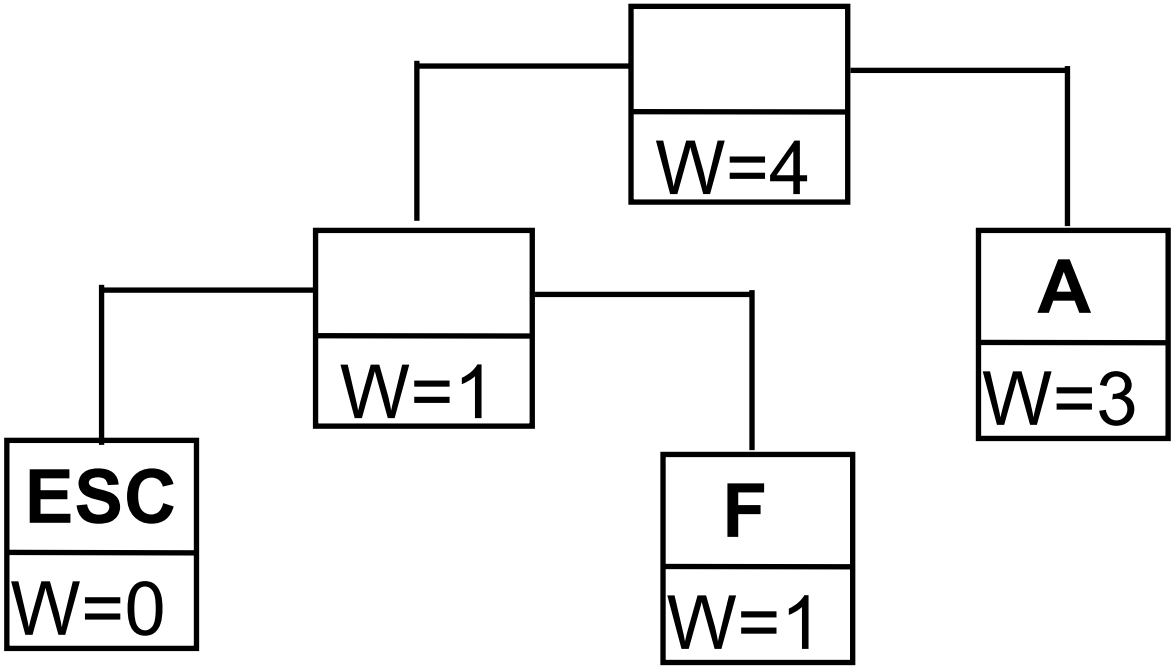
AAFAACEECC



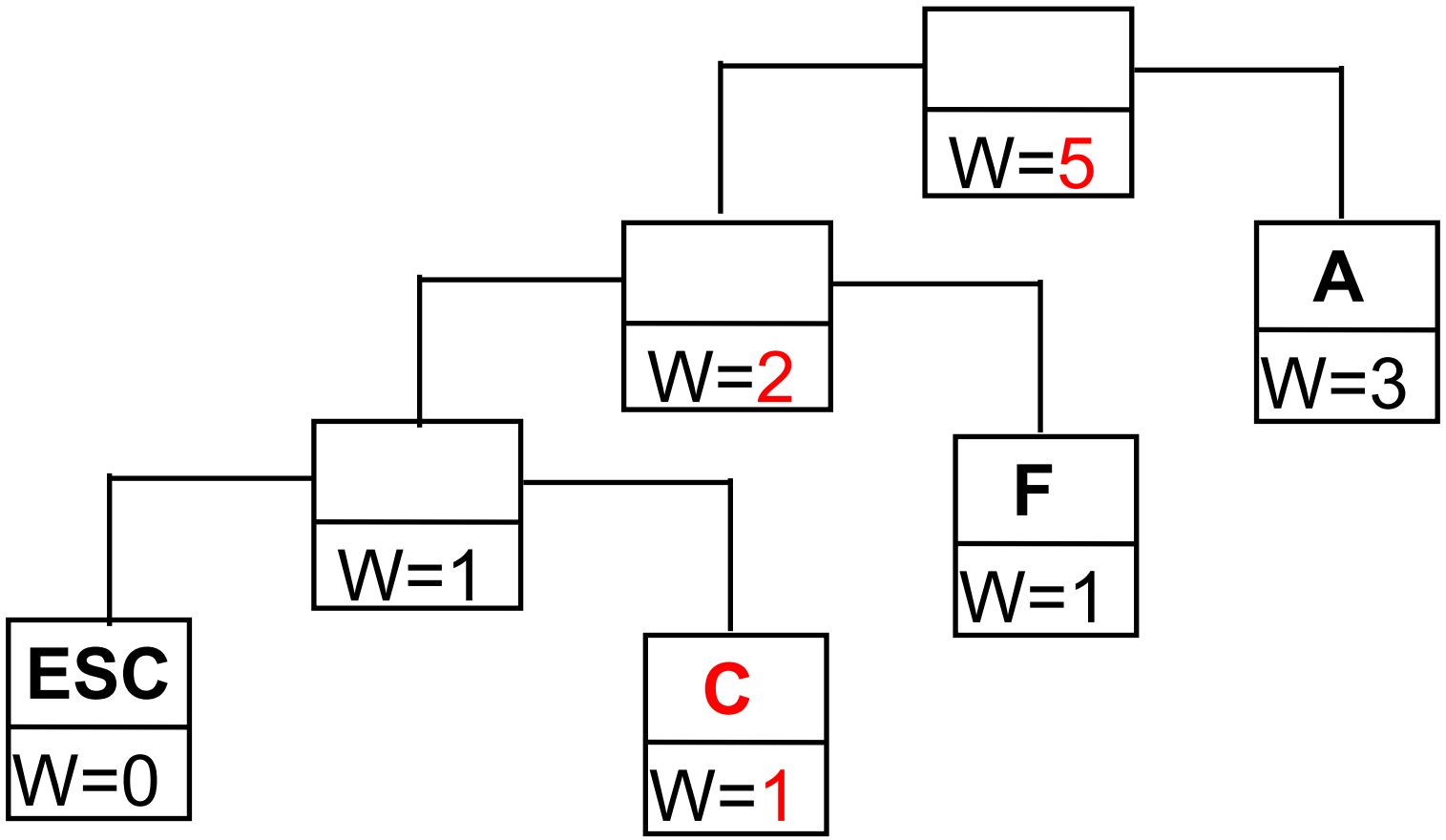
Для A передается код 1


W=4
W=1
A
W=3
ESC
W=0
F
W=1

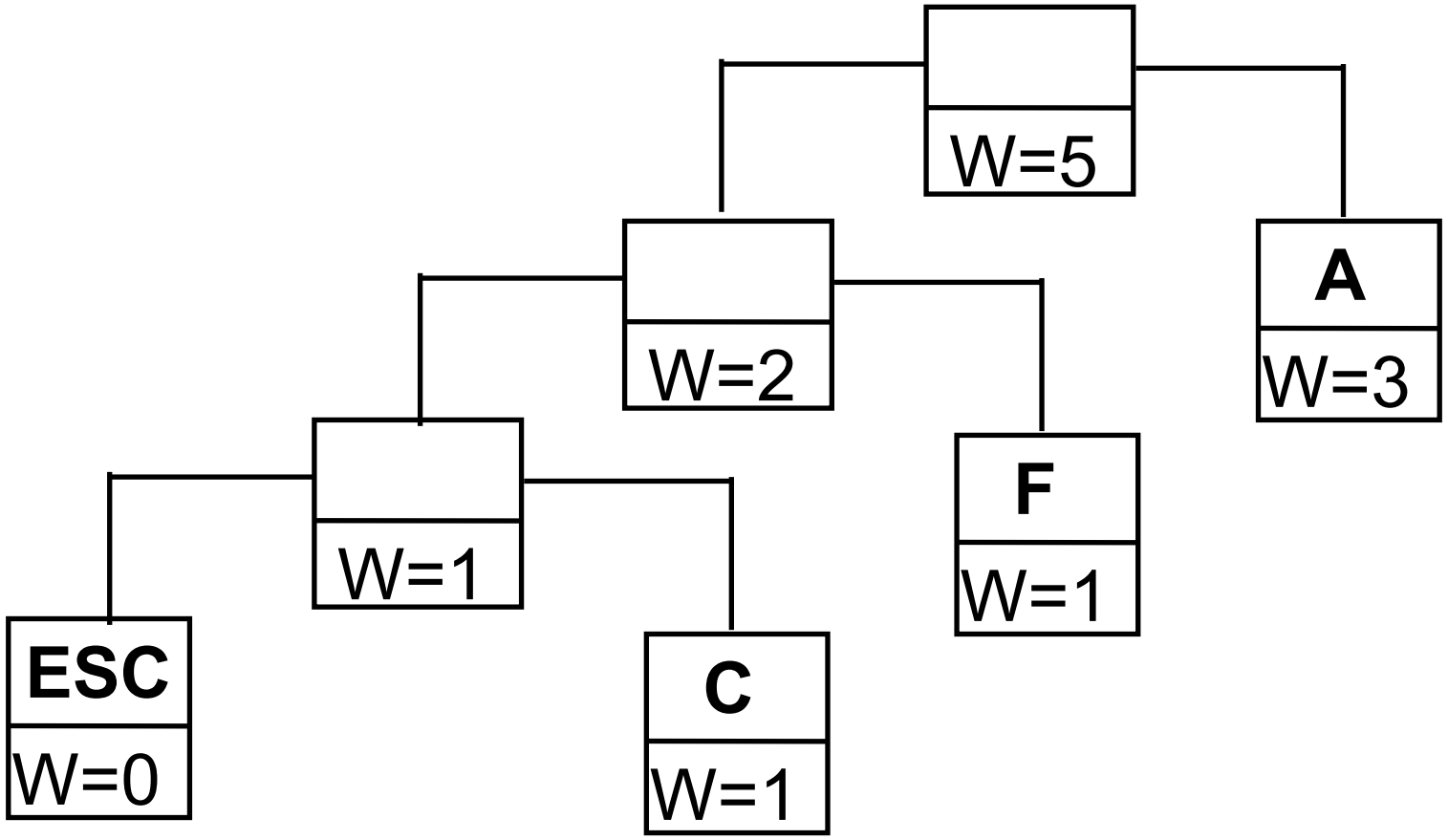
AAFACEECC



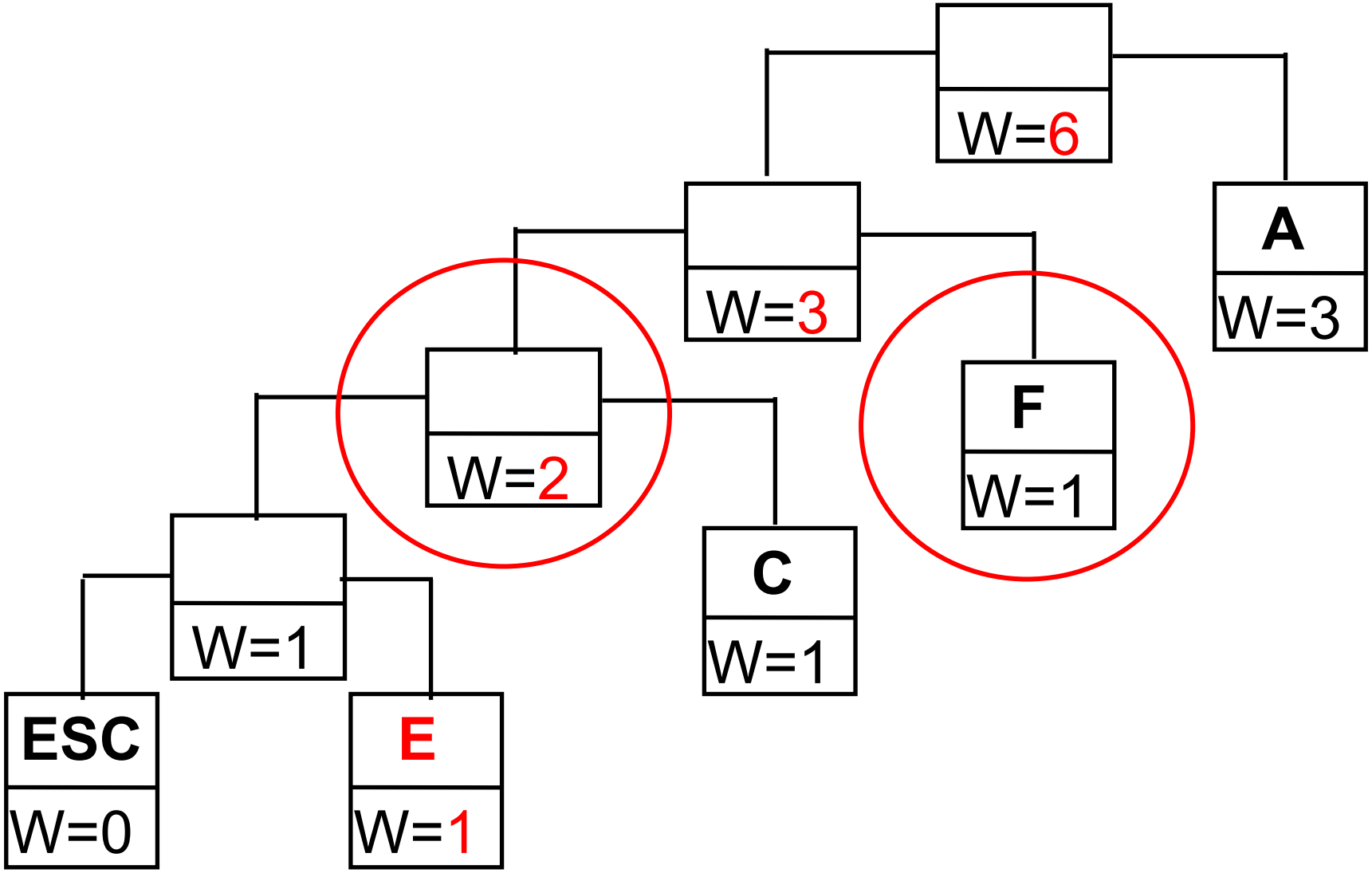
Для C передается код 00’C’


W=5
W=2
A
W=3
W=1
F
W=1
ESC
W=0
C
W=1

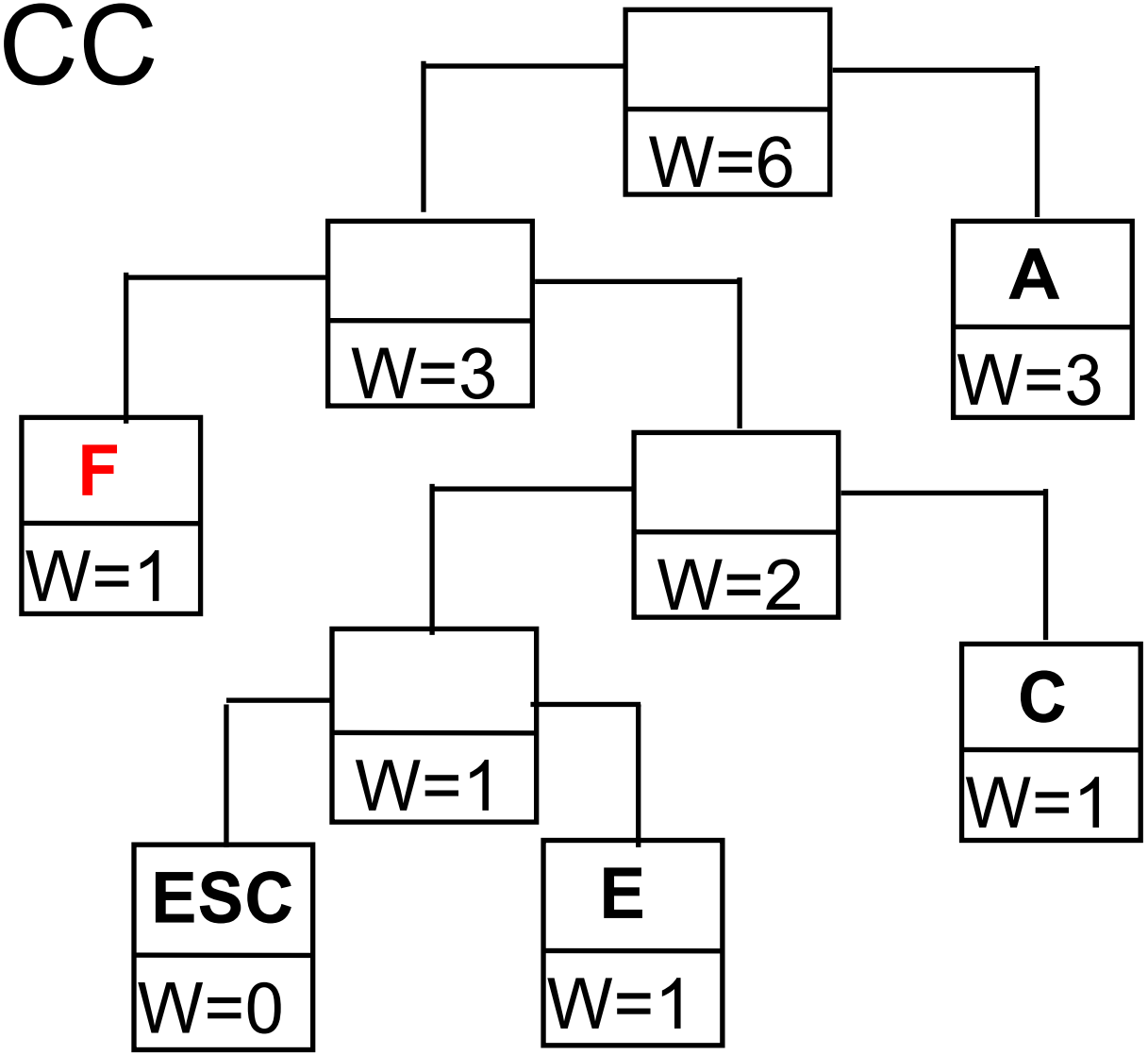
AAFACEECC



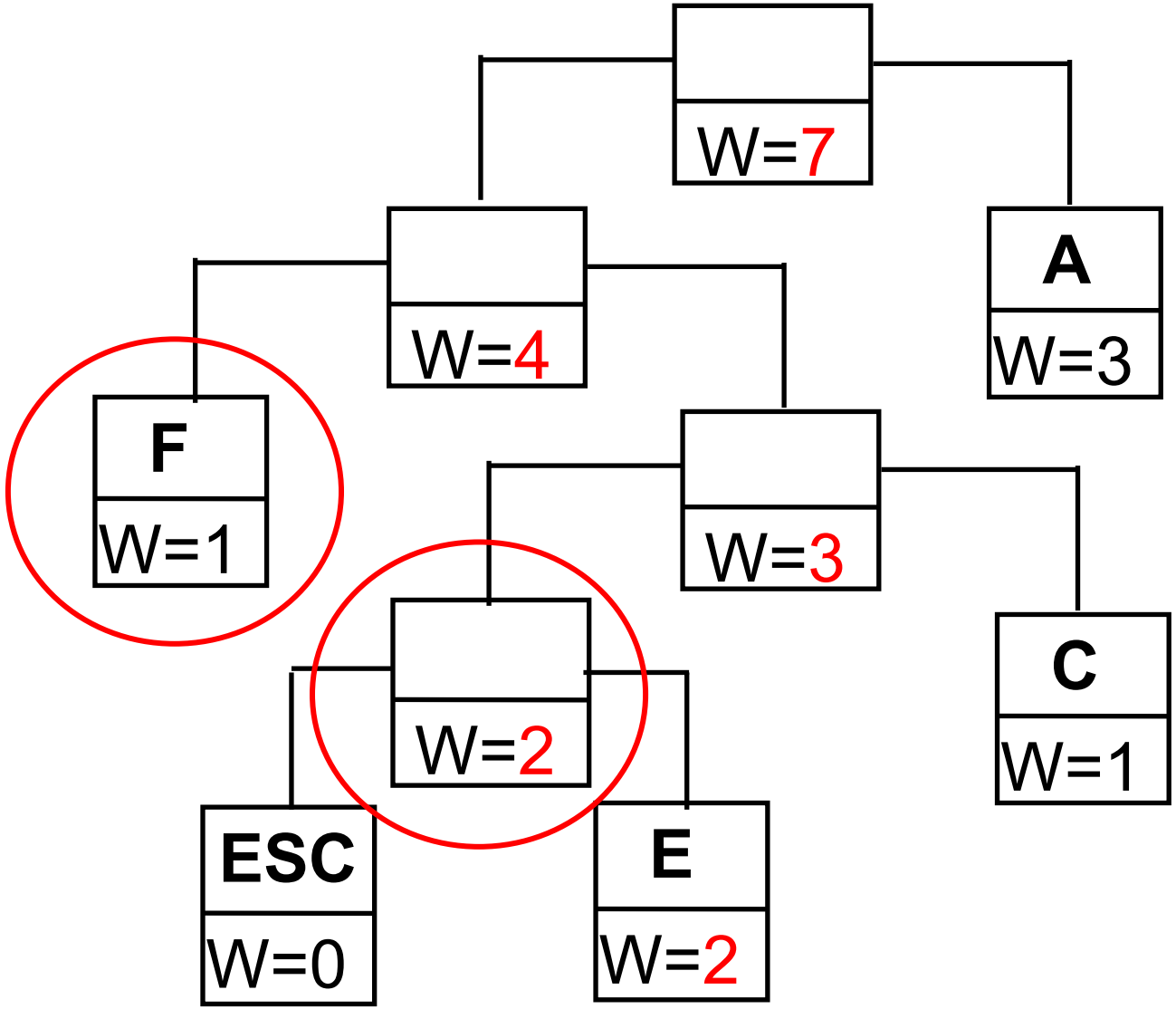
Для E передается код 000’E’

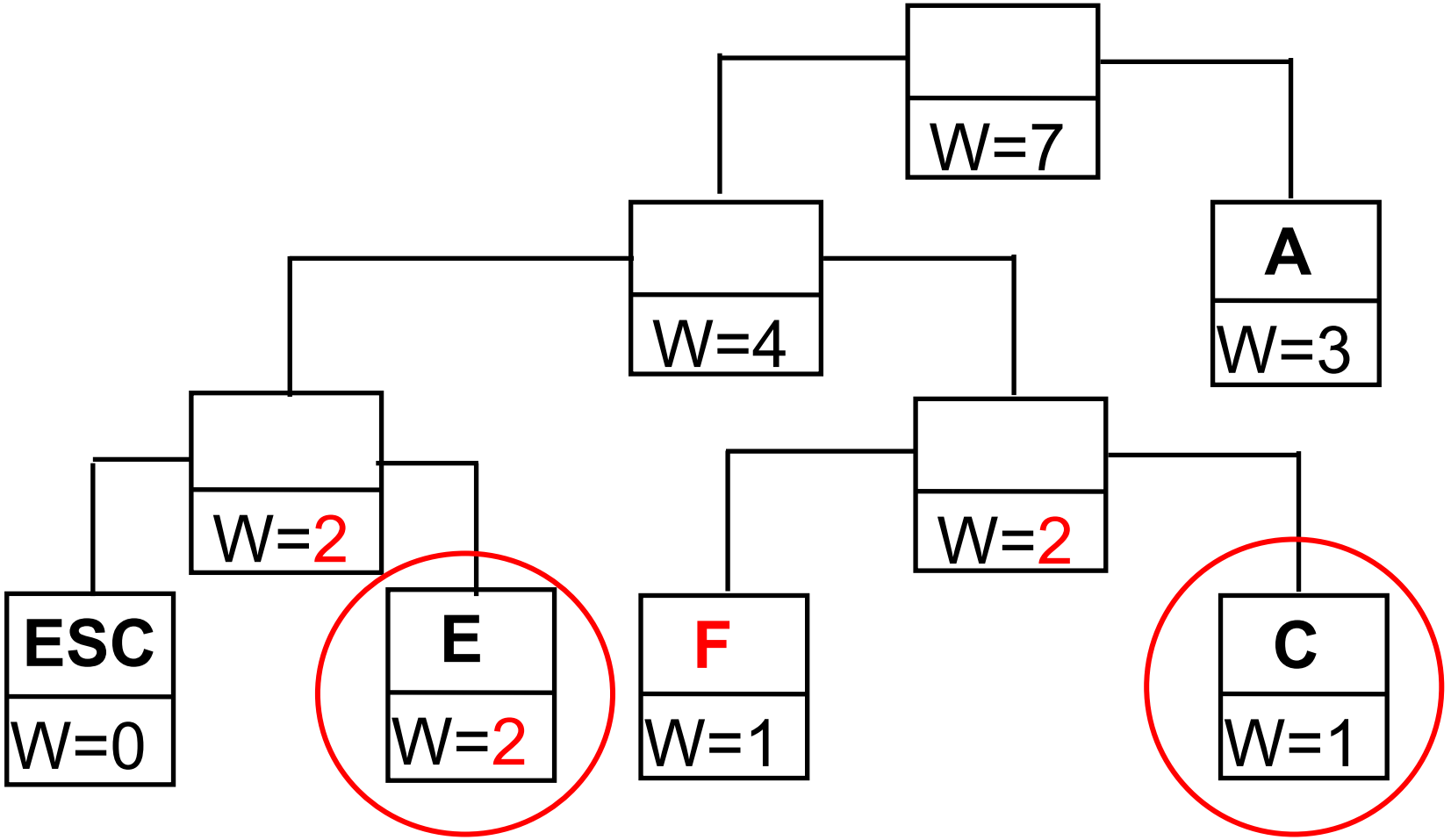

W=6
A
W=3
W=3
F
W=1
W=2
C
W=1
W=1
ESC
W=0
E
W=1

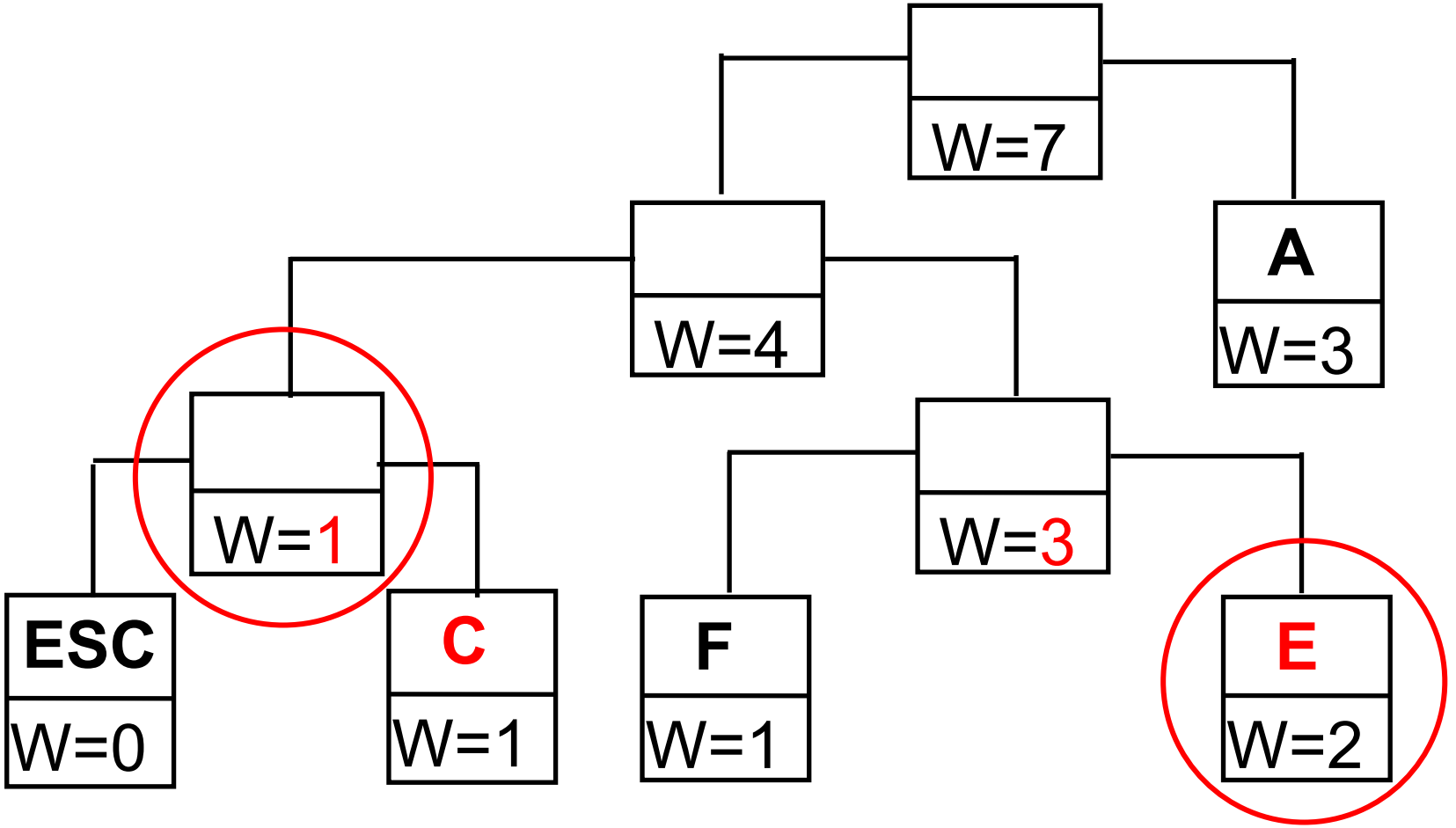
AAFACEECC

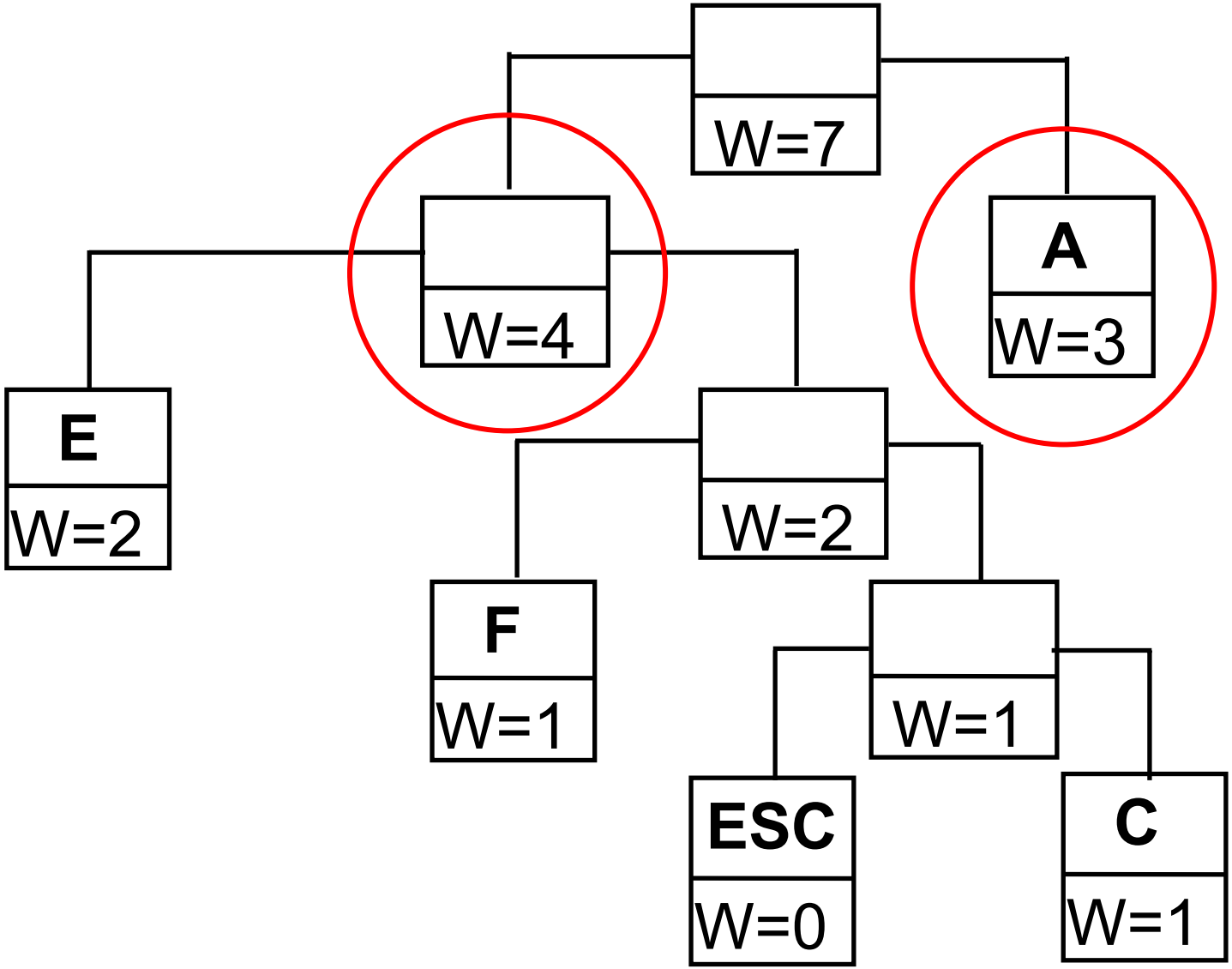


Для E передается код 0101


W=7
W=4
A
W=3
F
W=1
W=3
W=2
C
W=1
ESC
W=0
E
W=2

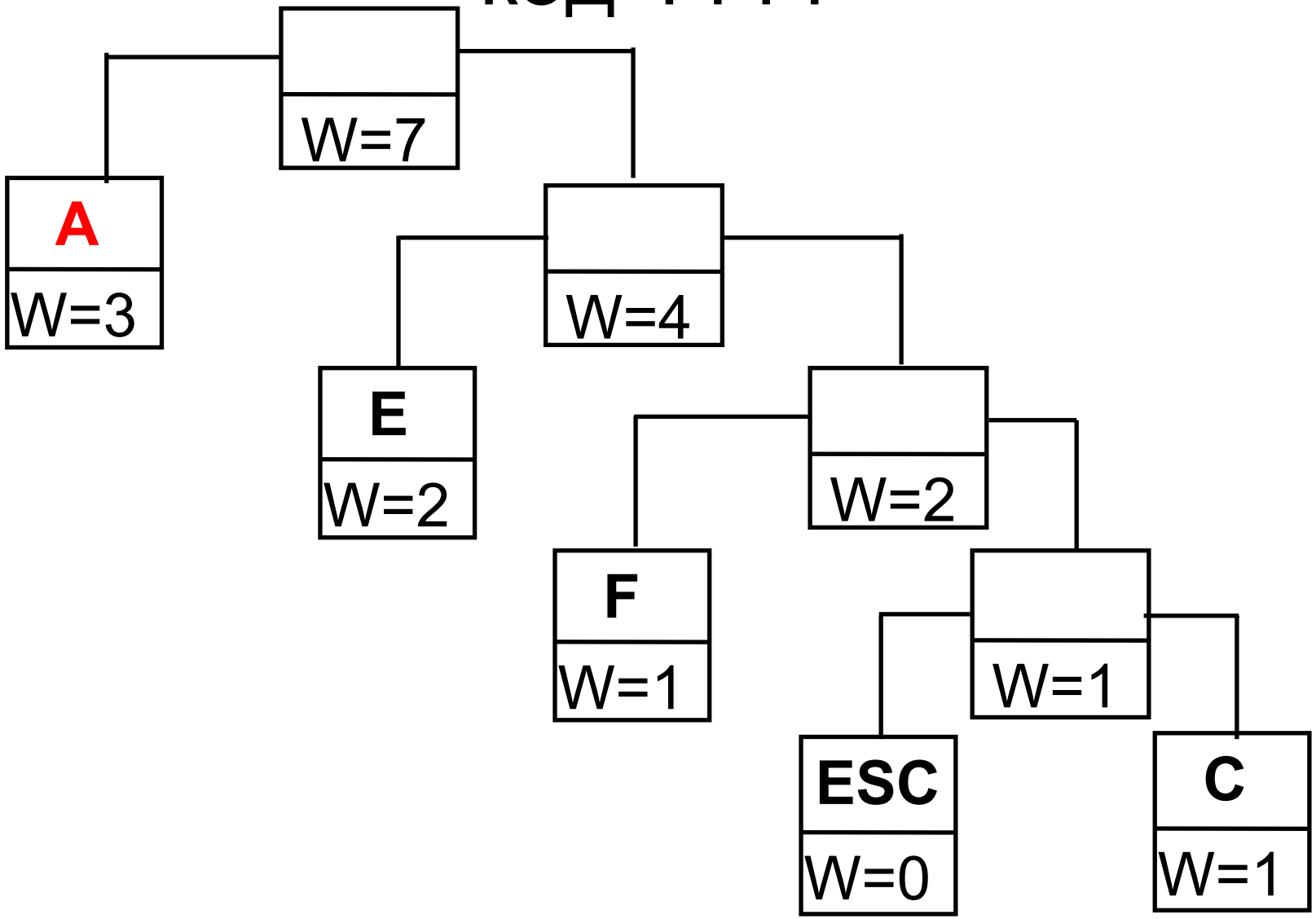

W=7
W=4
A
W=3
W=2
W=2
ESC
W=0
E
W=2
F
W=1
C
W=1

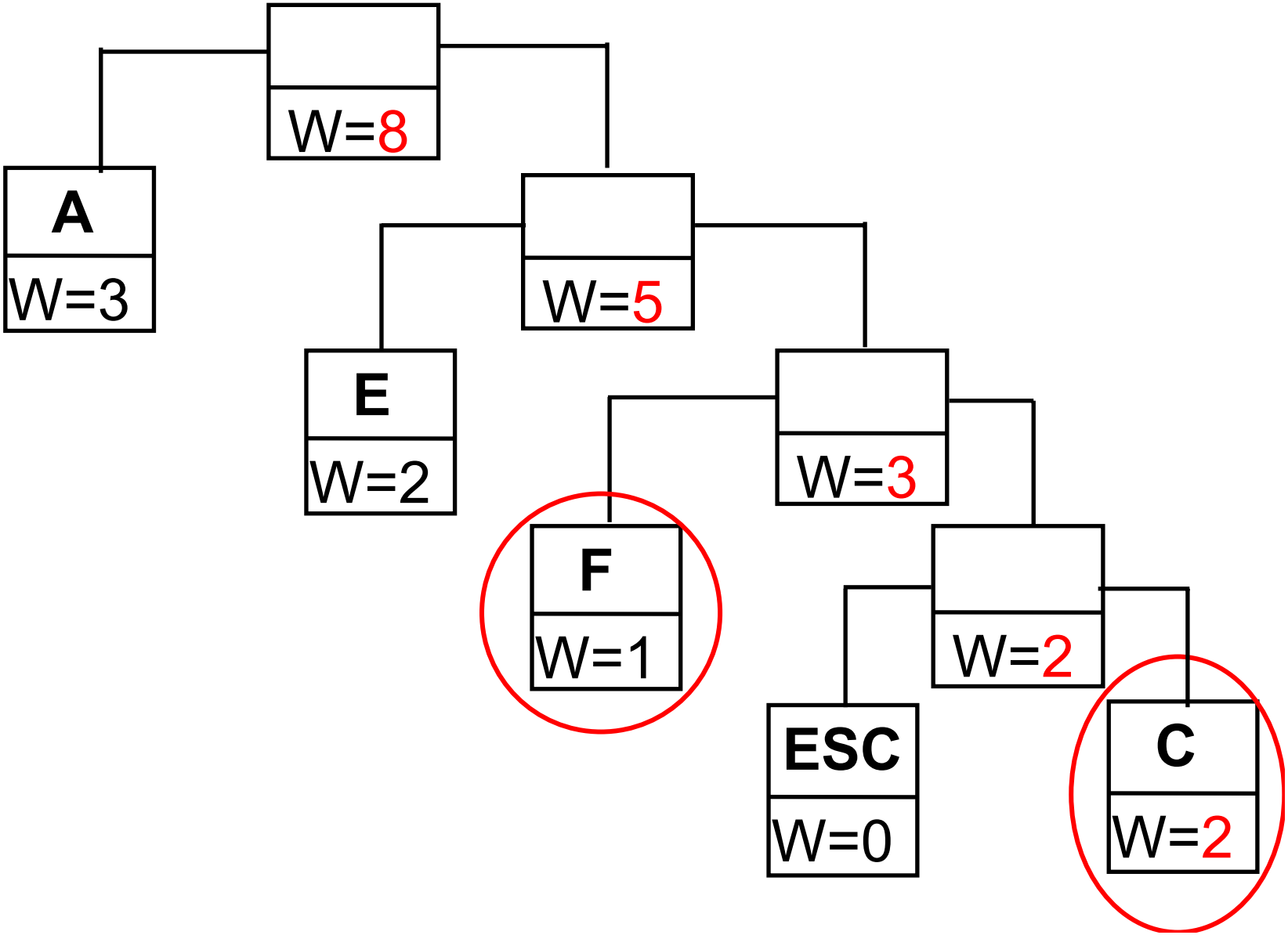

W=7
W=4
A
W=3
W=1
W=3
ESC
W=0
C
W=1
F
W=1
E
W=2

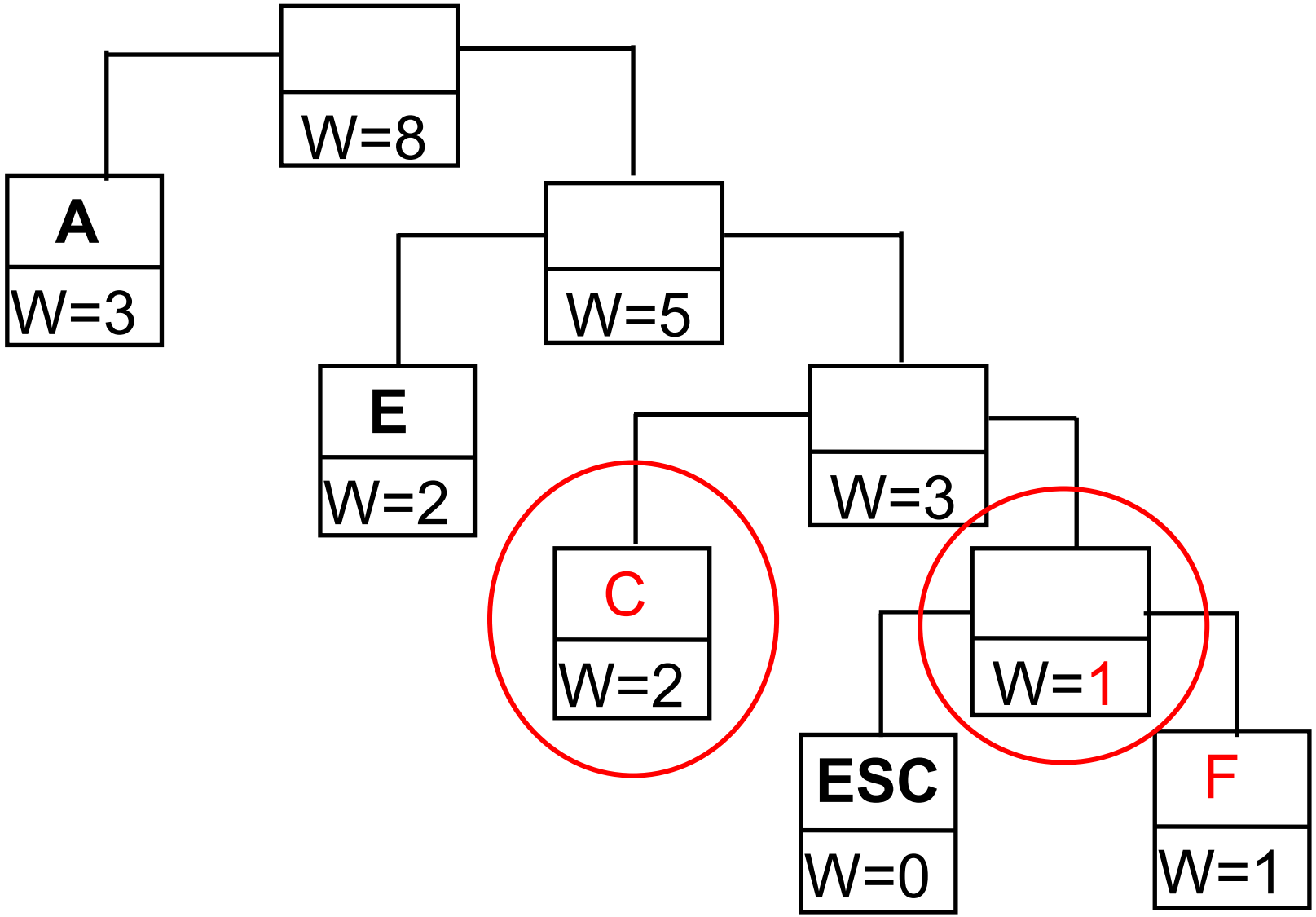

W=7
W=4
A
W=3
E
W=2
W=2
F
W=1
W=1
ESC
W=0
C
W=1

AAFACEEEC

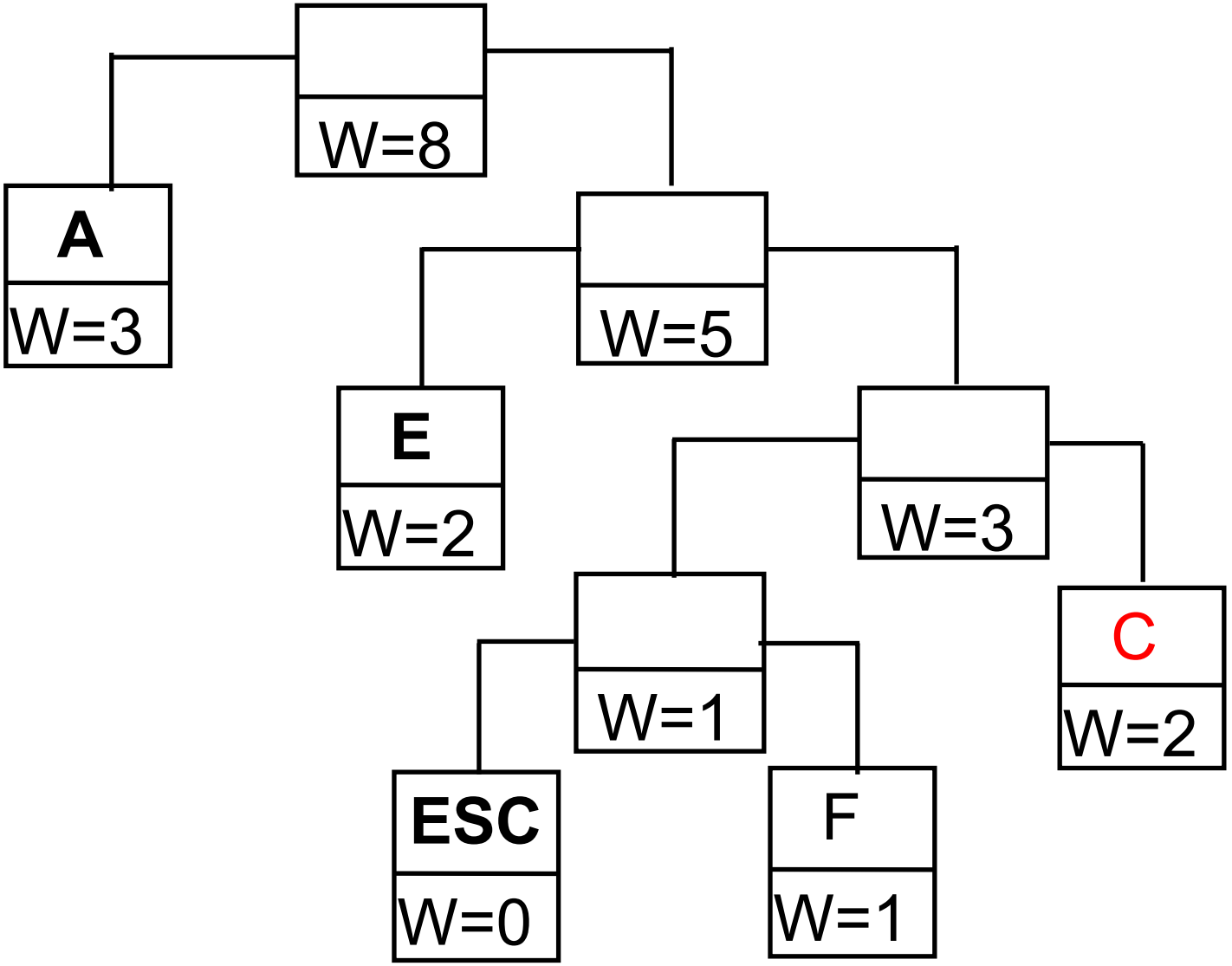
код 1111

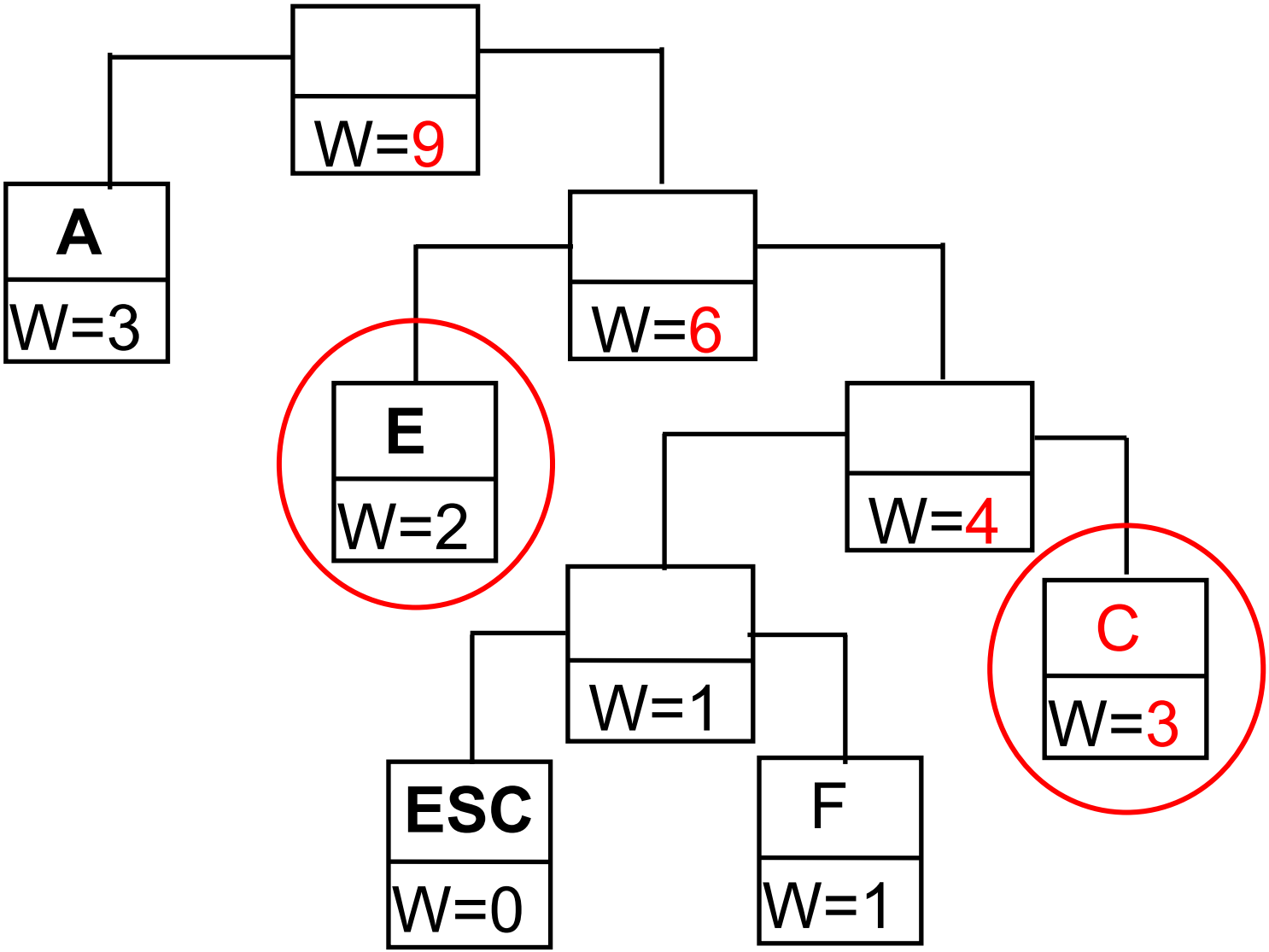


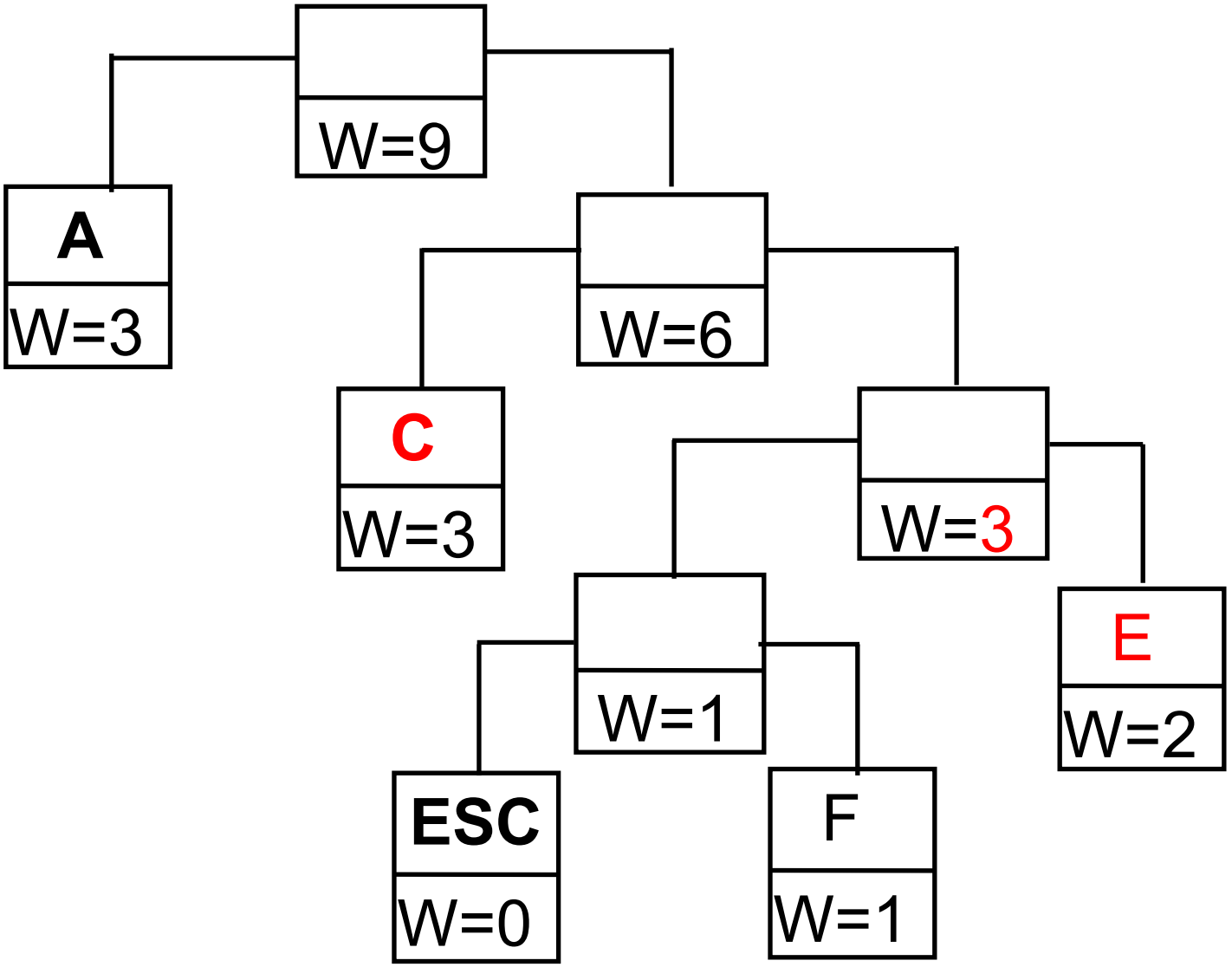

W=8
A
W=3
W=5
E
W=2
W=3
F
W=1
W=2
ESC
W=0
C
W=2


W=8
A
W=3
W=5
E
W=2
W=3
C
W=2
W=1
ESC
W=0
F
W=1

AAFACEECC код 111




W=9
A
W=3
W=6
E
W=2
W=4
W=1
C
W=3
ESC
W=0
F
W=1


W=9
A
W=3
W=6
C
W=3
W=3
W=1
E
W=2
ESC
W=0
F
W=1