Applied Algorithms

CSCI-B505 / INFO-I500

Lecture 13.

Huffman Codes

M. Oguzhan Kulekci

- Introduction to data compression
- Huffman Coding
  - Pre-Huffman with Shannon-Fano Codes
  - Huffman Coding
  - Implementation with heap

Efficient Transmission & Storage of Information

CARVING

WRITING

DIGITIZING

- Has been receiving interest for a long time, really :)
- Capabilities improve, but also the requirements, and demand
- How to do it most efficiently ?

WHAT IS DATA COMPRESSION ?




- **Data size always increases proportional to available resource !**
- **Thus, find ways to represent it as concise as possible.**

Remove the redundancy, and squeeze the data down to its information content (entropy) , which is analogous to a vacuum storage bag.

MAIN STEPS OF DATA COMPRESSION ?

1. MODELING

- Find a good way to describe your data, which helps to make implicit redundancies explicit.
- Very important since we can compress the data as much as we understand it!

2. ENTROPY CODING

- Encode your transformed data with a chosen entropy coder (Huffman, Arithmetic, ANS).
- The effect of entropy coders with respect to modeling is less significant !

Today we will focus on a widely used entropy coding technique :
The Huffman Codes

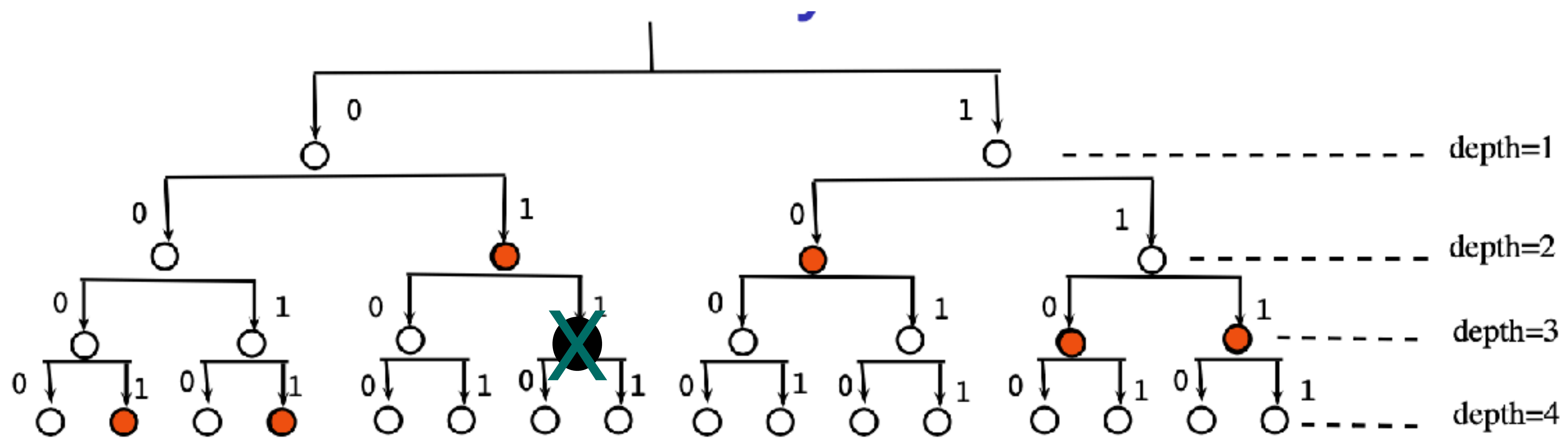
Main Idea

- Assign a binary codeword to each symbol
- Represent frequent symbols with shorter codewords
- How to make such an assignment ?

`abracadabra`

- We have five symbols so need 3-bits per each with fixed-length coding
- Frequencies: 5 a, 2 b, 2 r, 1 c, 1 d
- To make it compact, say $a \rightarrow 0,\ \ b \rightarrow 1,\ \ c \rightarrow 00,\ \ d \rightarrow 01,\ \ r \rightarrow 10$
- Then, abracadabra = 011000000101100, much better than 11x3=33 bit fixed length representation.
- But, how to decode it !!! Codeword boundaries are lost.
- So, assigning codewords is not only a matter of frequency !

Prefix-Free Codes



- Let 's mark the codewords on a binary tree.
- What happens if the assigned codewords are not leaves? **Ambiguity !!!**
- Thus, no codeword should appear as a prefix of another.
- Being **prefix free** guarantees unique decodability.

Now, the question is how to assign prefix-free codewords respecting the frequency of the symbols ?

Initial Attempts: Shannon - Fano Coding

- Shannon-1948 and Fano-1949, independently
- SF does not guarantee optimal codes (unlike Huffman as we will see)
- SF codes guarantee that the codeword of each symbol is at most 1 bit larger than its entropy *(out of our scope today)*

Shannon's Way

Symbol	A	B	C	D	E
Probabilities	0.385	0.179	0.154	0.154	0.128
$-\log_2 p_i$	1.379	2.480	2.700	2.700	2.963
Word lengths $\lceil -\log_2 p_i \rceil$	2	3	3	3	3
Codewords	00	010	011	100	101

Figure from https://en.wikipedia.org/wiki/Shannon–Fano_coding

Fano’s Way

Symbol	A	B	C	D	E
Probabilities	0.385	0.179	0.154	0.154	0.128
First division	0		1		
Second division	0	1	0	1	
Third division				0	1
Codewords	00	01	10	110	111

Figure from https://en.wikipedia.org/wiki/Shannon–Fano_coding

Shannon - Fano Example

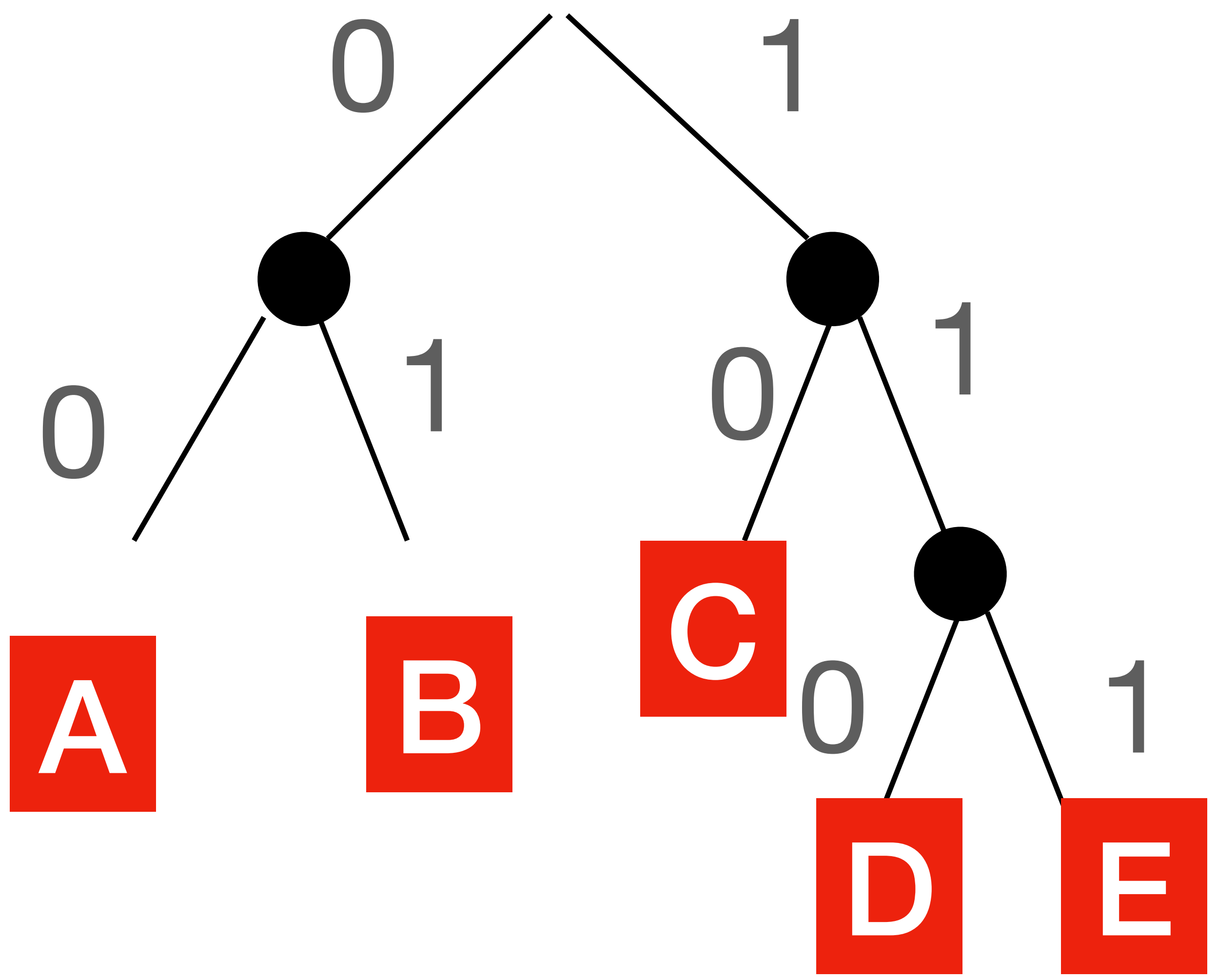
Symbol:	A	B	C	D	E
Frequency:	15	7	6	6	5

- Sort symbols in decreasing frequency,
- Split into two most balanced partitions
- Add 0 to the codes of the lower part, and 1 to upper part
- Recurse same on the partitions until they become single symbols

Shannon - Fano Example

15 - 24   **22 - 17**   28 - 11   34 - 5

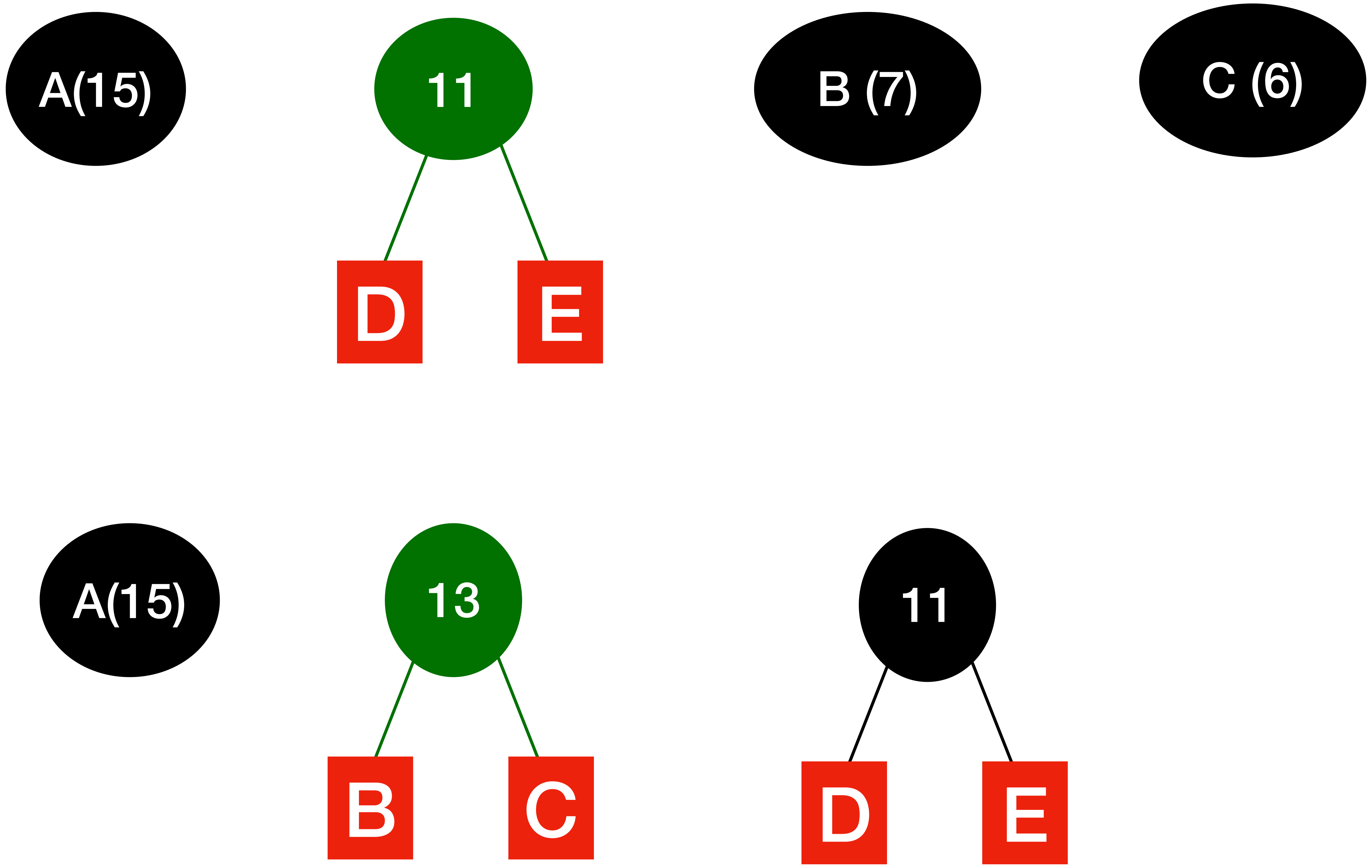
Symbol:	A	B	C	D	E
Frequency:	**15**	**7**	**6**	**6**	**5**
	15 (0)	**7 (0)**	**6 (1)**	**6 (1)**	**5 (1)**
	15 (00)	**7 (01)**	**6 (10)**	**6 (11)**	**5 (11)**
				6 (110)	**5 (111)**



- Shannon-Fano is a top-down approach
- Optimality is not guaranteed !
- Huffman solved the optimality issue with a bottom-up approach

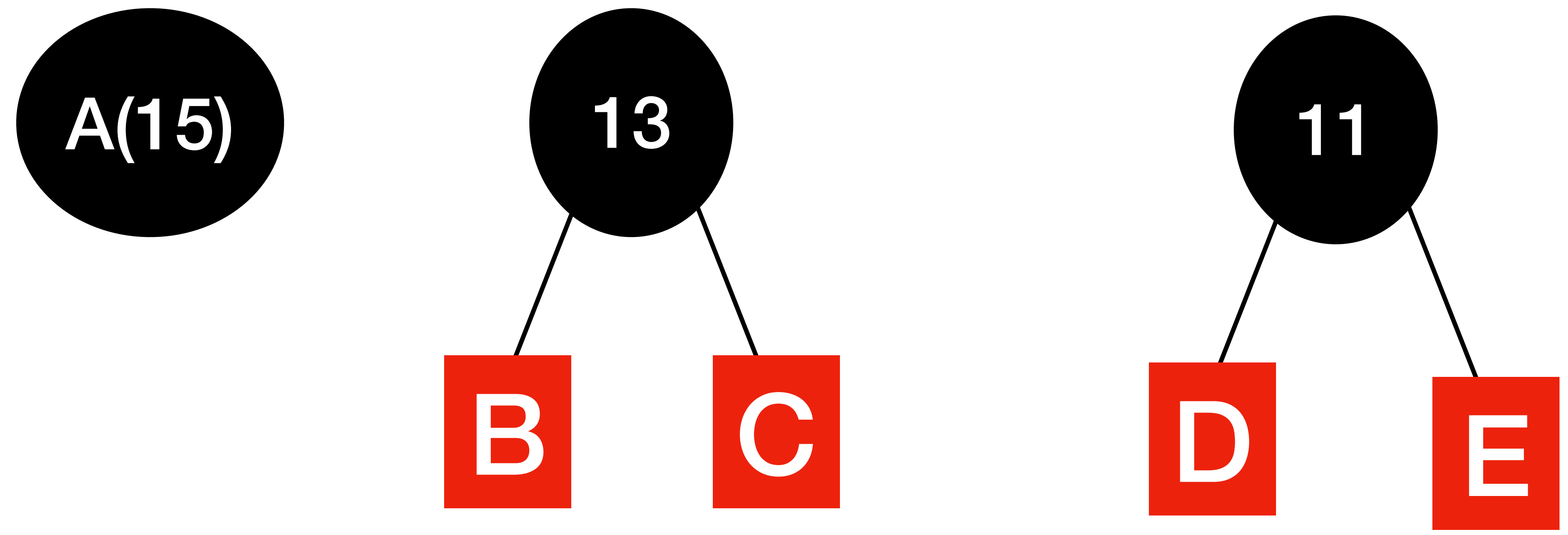
Huffman

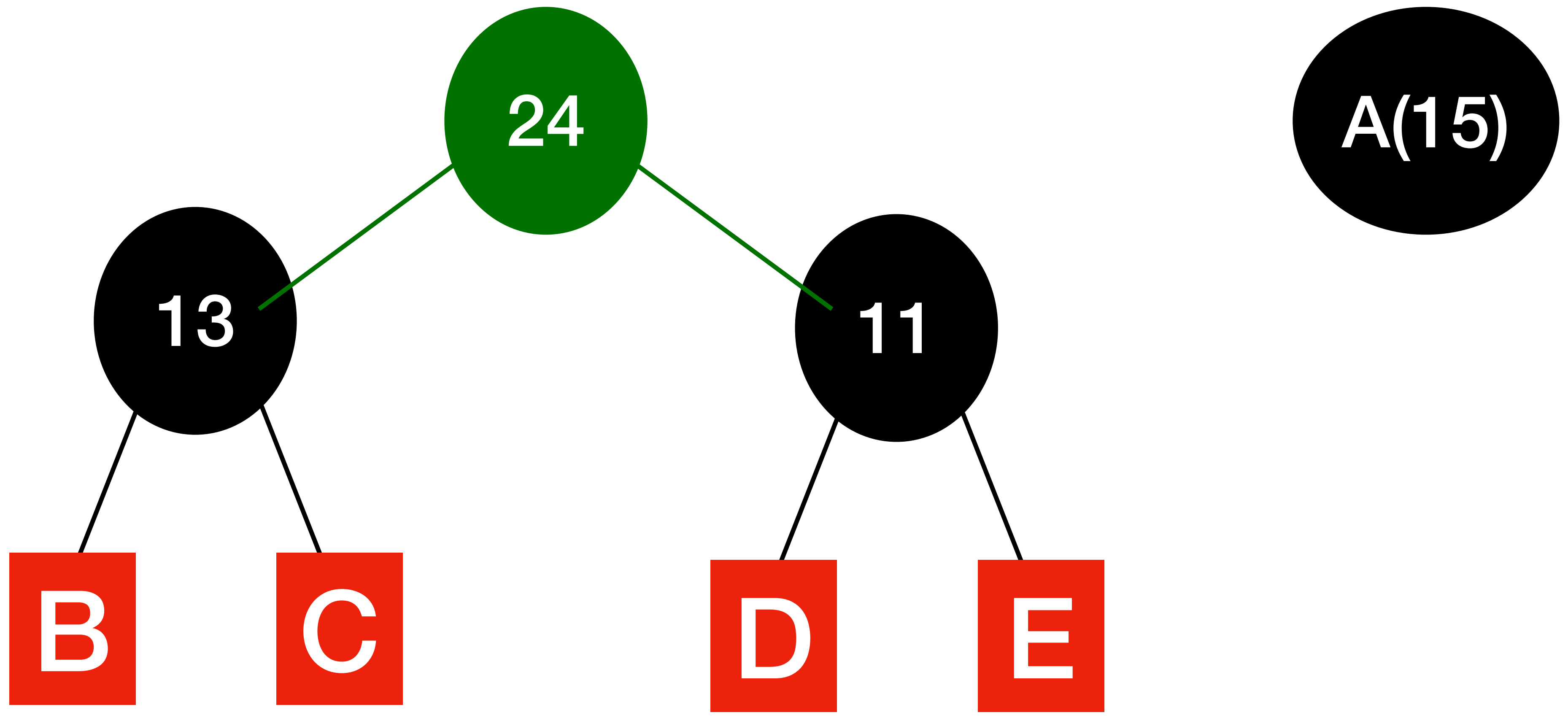
Symbol:	A	B	C	D	E
Frequency:	15	7	6	6	5

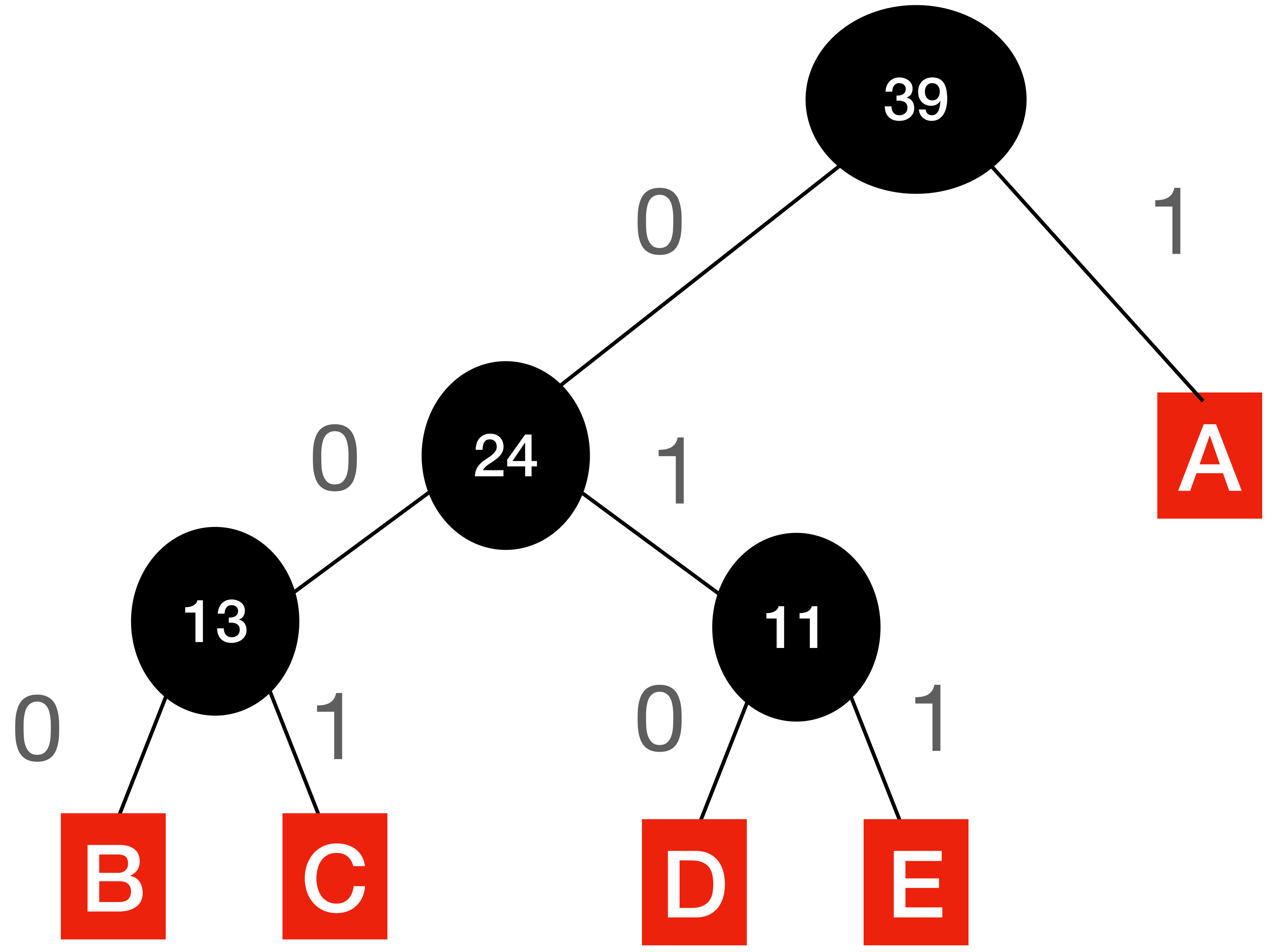


Huffman

Symbol:	A	B	C	D	E
Frequency:	**15**	**7**	**6**	**6**	**5**





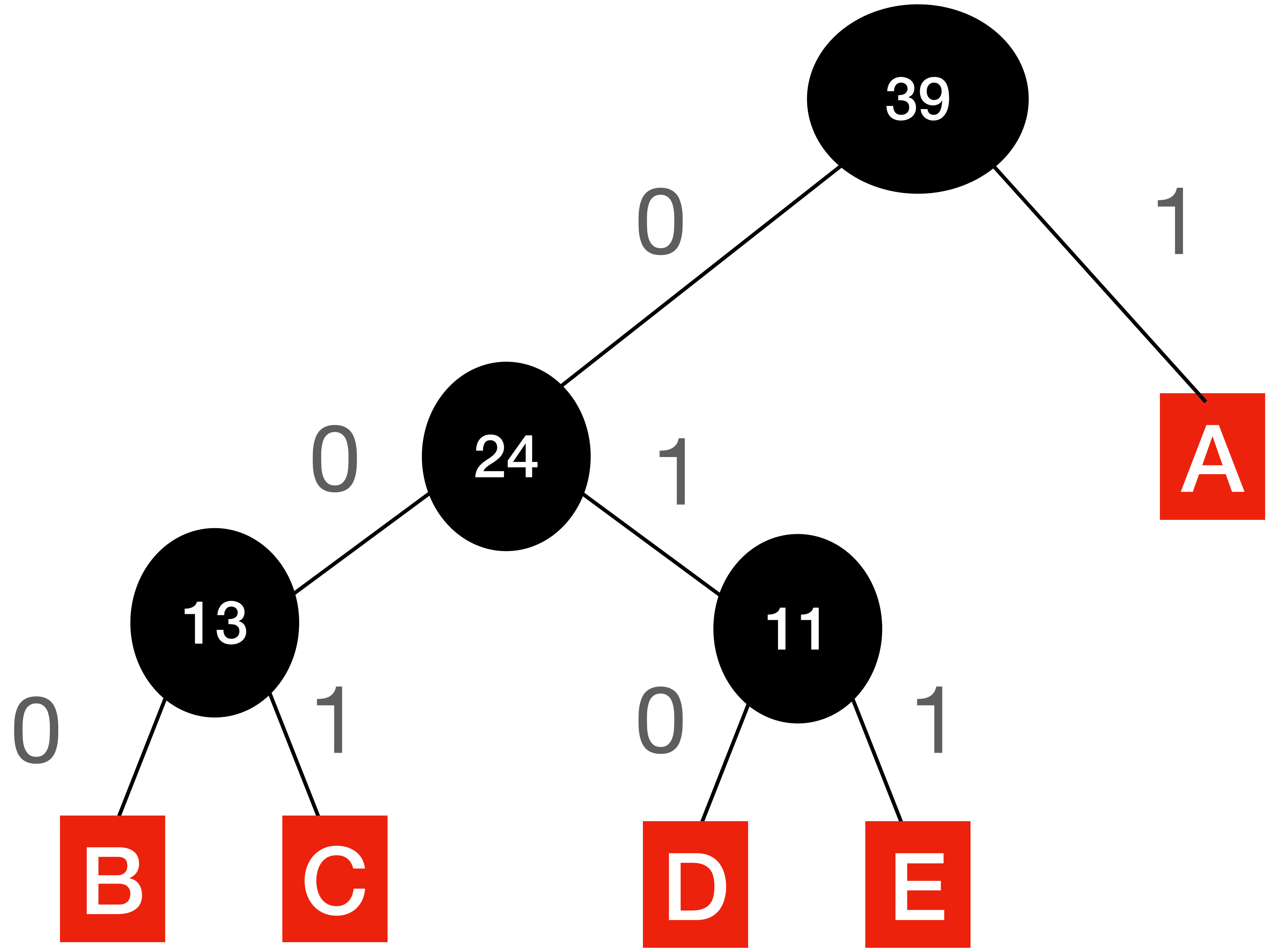


Huffman

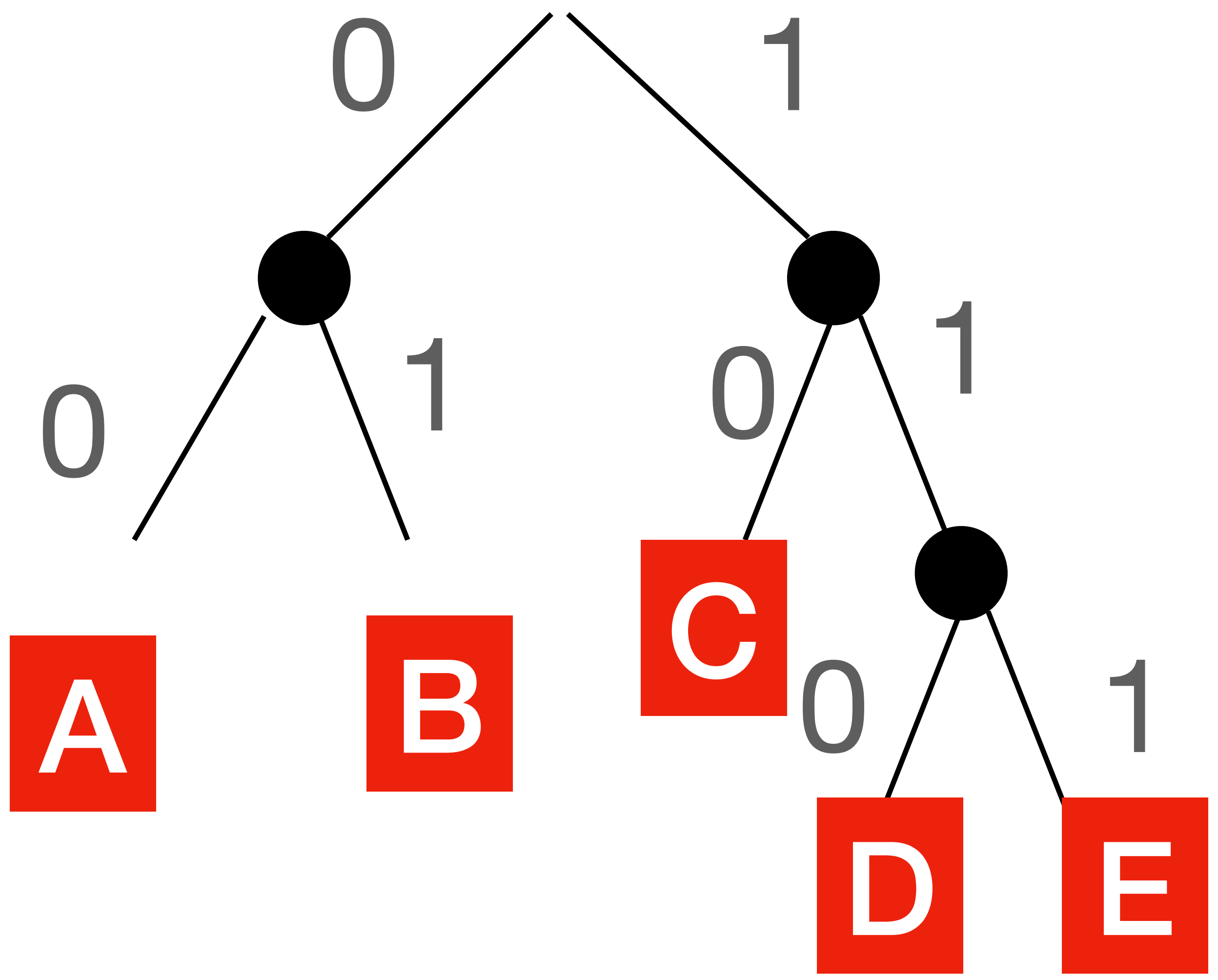
0010101000011100100 0

Symbol:	A	B	C	D	E
Frequency:	**15**	**7**	**6**	**6**	**5**
Huffman	**1**	**000**	**001**	**010**	**011**
Shannon-Fano	**00**	**01**	**10**	**110**	**111**

$$15 \cdot 1 + 24 \cdot 3 = 87$$



$$28 \cdot 2 + 11 \cdot 3 = 89$$



Implementing Huffman

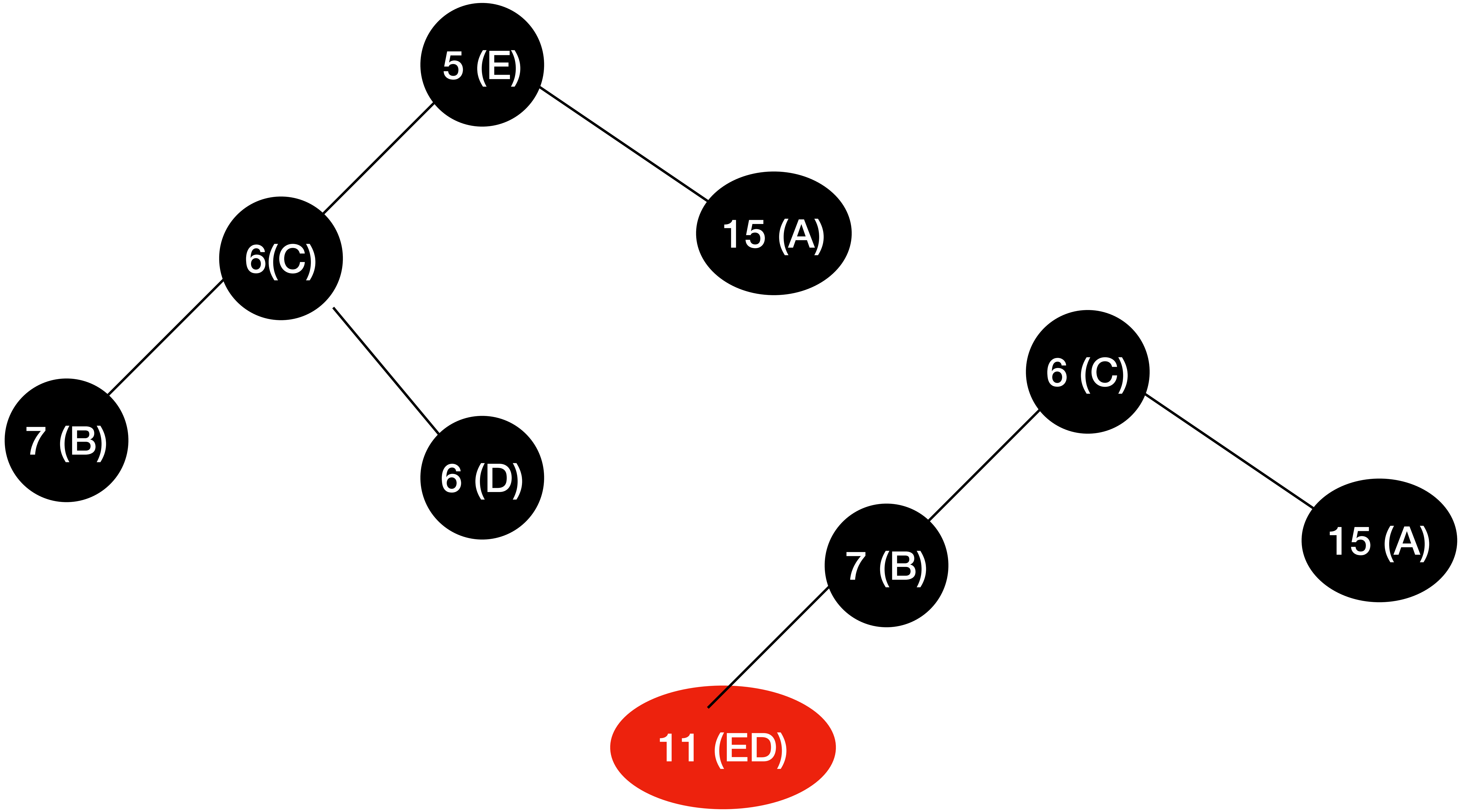
Symbol:	A	B	C	D	E
Frequency:	**15**	**7**	**6**	**6**	**5**

1. Create a min-heap from the frequency counts.

2. Extract the minimum from the heap, and append 0 to their codewords
3. Extract the minimum from the heap again, and append 1 to their codewords
4. Merge these two least frequent symbols into a new symbol, whose frequency is the sum of these symbols.
5. Insert the new symbol into the min-heap with its corresponding frequency count.
6. Go to step 2 until there remains only one node in the min-heap.
7. *Reverse all codewords generated! Why?*

Implementing Huffman

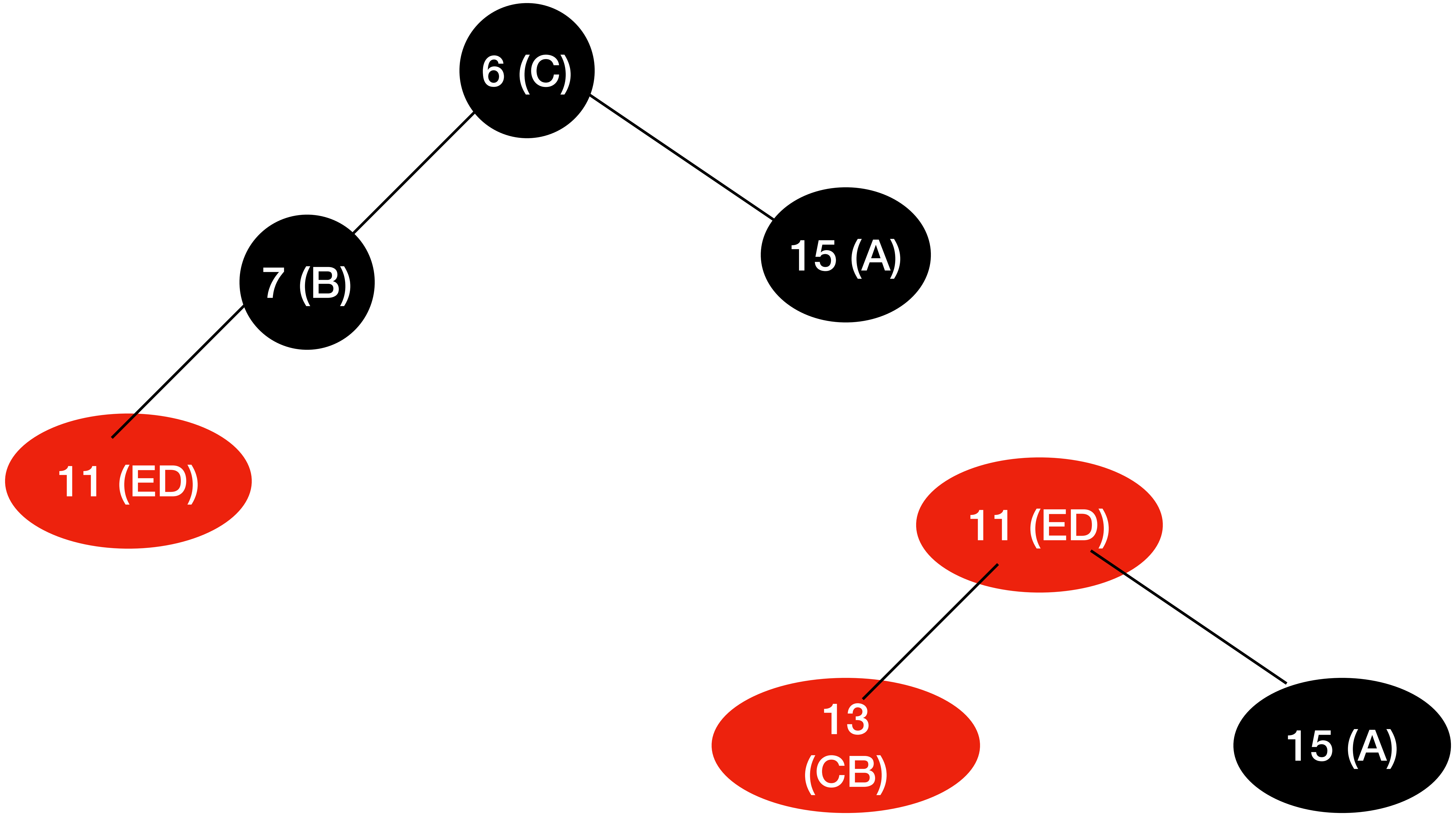
Symbol:	A	B	C	D	E
Frequency:	**15**	**7**	**6**	**6**	**5**



Symbol:	Codeword
A	
B	
C	
D	**1**
E	**0**

Implementing Huffman

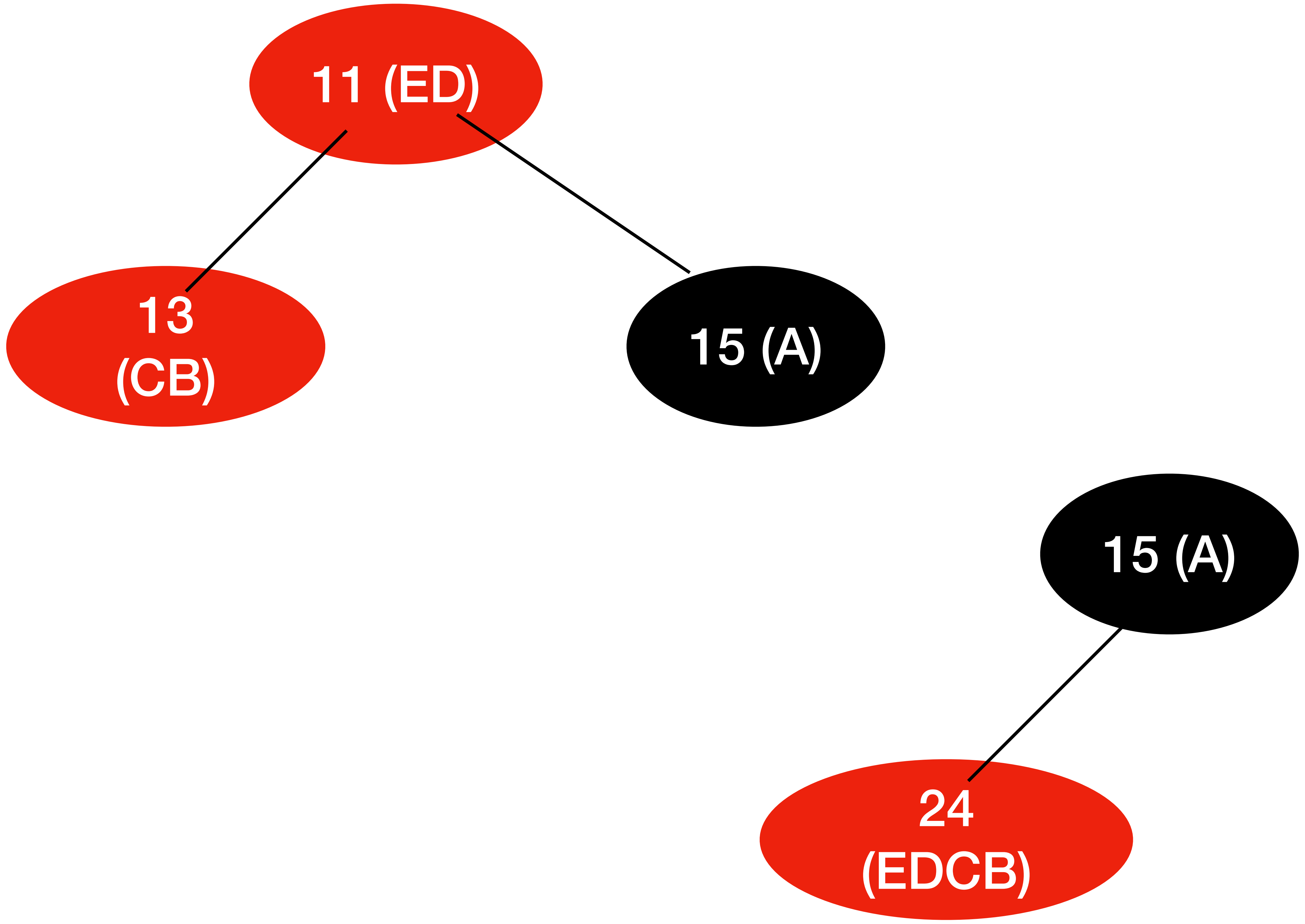
Symbol:	A	B	C	D	E
Frequency:	**15**	**7**	**6**	**6**	**5**



Symbol:	Codeword
A	
B	**1**
C	**0**
D	**1**
E	**0**

Implementing Huffman

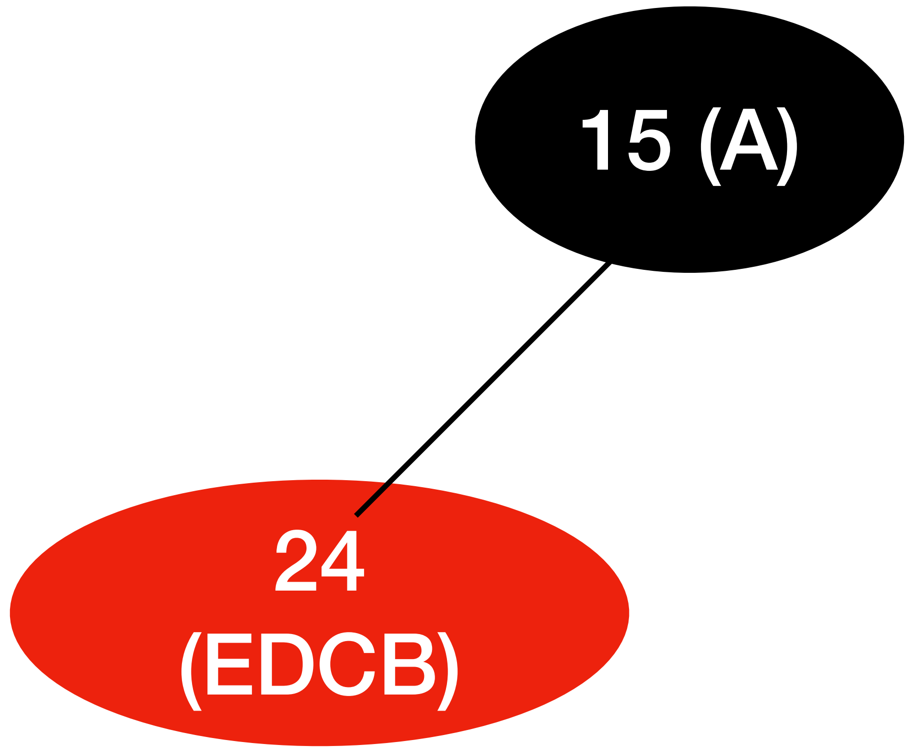
Symbol:	A	B	C	D	E
Frequency:	**15**	**7**	**6**	**6**	**5**



Symbol:	Codeword
A	
B	**11**
C	**01**
D	**10**
E	**00**

Implementing Huffman

Symbol:	A	B	C	D	E
Frequency:	**15**	**7**	**6**	**6**	**5**



Symbol:	Codeword
A	**0**
B	**111**
C	**011**
D	**101**
E	**001**

Decoding Huffman

- Receiver needs additional information to correctly decode the compressed data.
- Many different ways to achieve this (out of our scope today…)
- All need to maintain a header before the actual Huffman codes to let receiver generate the same codes of the sender.

Reading assignment

- Read the chapter 4.8 from Kleinberg&Tardos, chapter 16.3 from Cormen.