Multimedia
Lecture 3

Dr. Mona M.Soliman

Faculty of Computers and Artificial Intelligence

Cairo University

Feb 2020

Variable Length Coding

Variable Length Coding (VLC)

abaacaadaa *(10 Symbols)*

P(a)=7/10, P(b)=1/10, P(c)=1/10, P(d)=1/10

Binary System assigns 2 Bits for Each Symbol

1

Symbol	Count	Code	Code Length	Total Bits
a	7	00	2	14
b	1	01	2	2
c	1	10	2	2
d	1	11	2	2

00010000100000110000

Symbols Can be stored in 20 Bits

We can Use Variable Length Codes for different Symbols

2

Symbol	Count	Code	Code Length	Total Bits
a	7	0	1	7
b	1	10	2	2
c	1	110	3	3
d	1	111	3	3

010001100011100

Symbols Can be stored In 15 Bits

Variable Length Coding (VLC)

abaacaadaa *(10 Symbols)*

P(a)=7/10, P(b)=1/10, P(c)=1/10, P(d)=1/10

Another Variable Length Codes for different Symbols

3

Symbol	Count	Code	Code Length	Total Bits
a	7	0	1	7
b	1	1	1	1
c	1	01	2	2
d	1	00	2	2

010001000000

Symbols Can be stored in 12 Bits

Another Variable Length Codes for different Symbols

4

Symbol	Count	Code	Code Length	Total Bits
a	7	10	2	14
b	1	0	1	1
c	1	111	3	3
d	1	110	3	3

100101011110101101010

Symbols Can be stored In 21 Bits

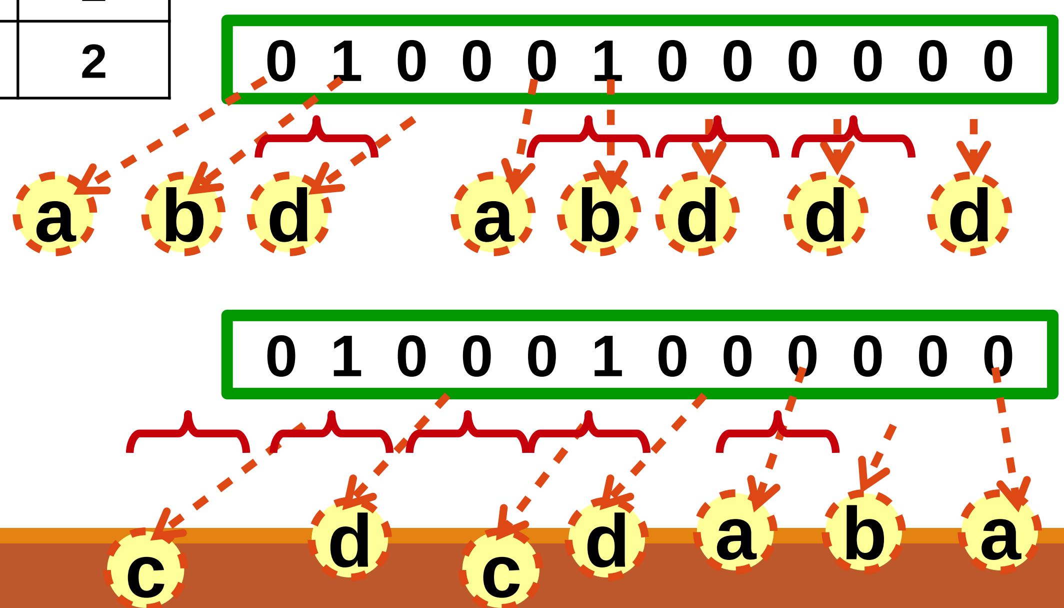
Confusing Codes

abaacaadaa *(10 Symbols)*

P(a)=7/10, P(b)=1/10, P(c)=1/10, P(d)=1/10

3

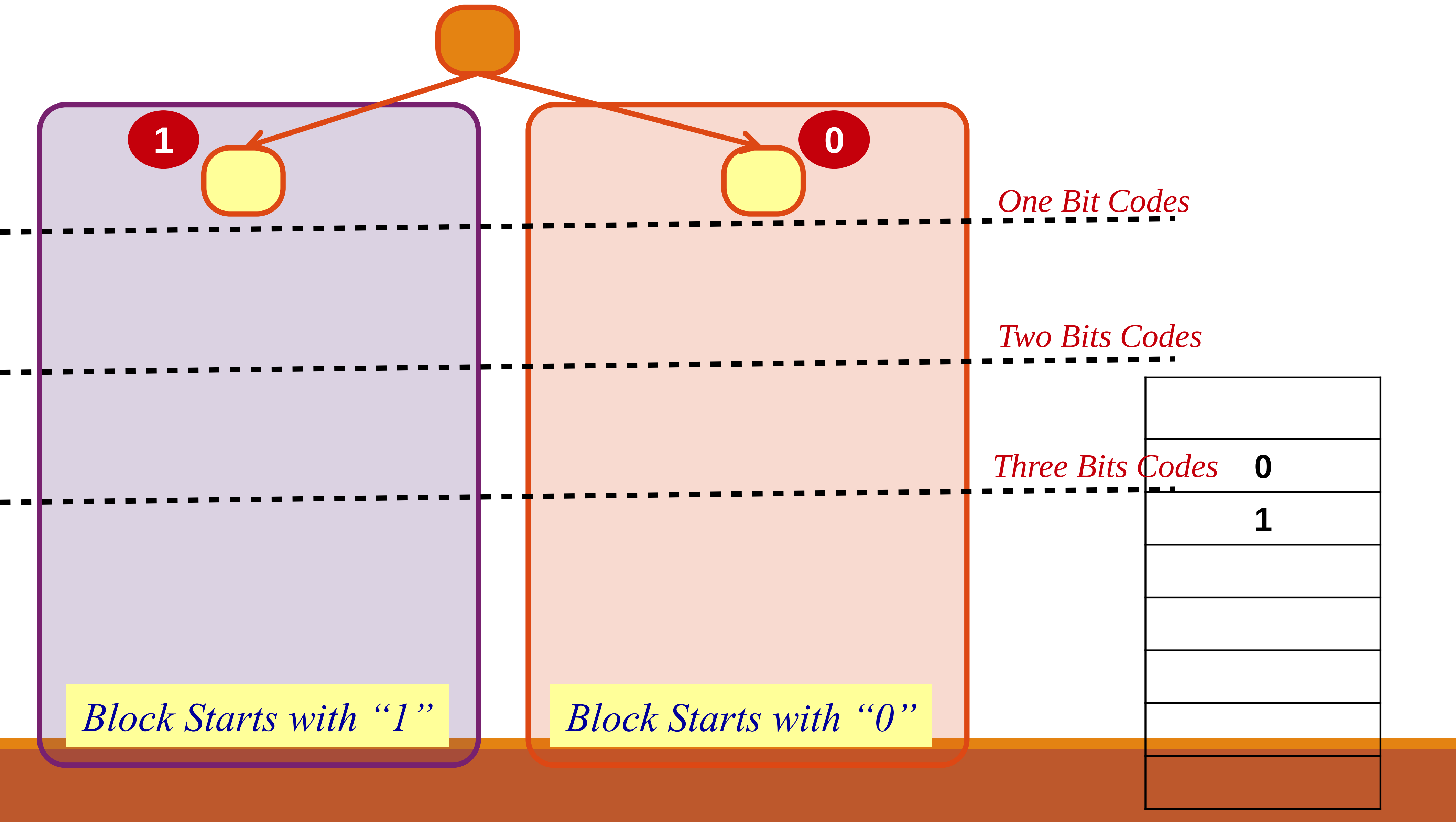
Symbol	Count	Code	Code Length	Total Bits
a	**7**	**0**	**1**	**7**
b	**1**	**1**	**1**	**1**
c	**1**	**01**	**2**	**2**
d	**1**	**00**	**2**	**2**



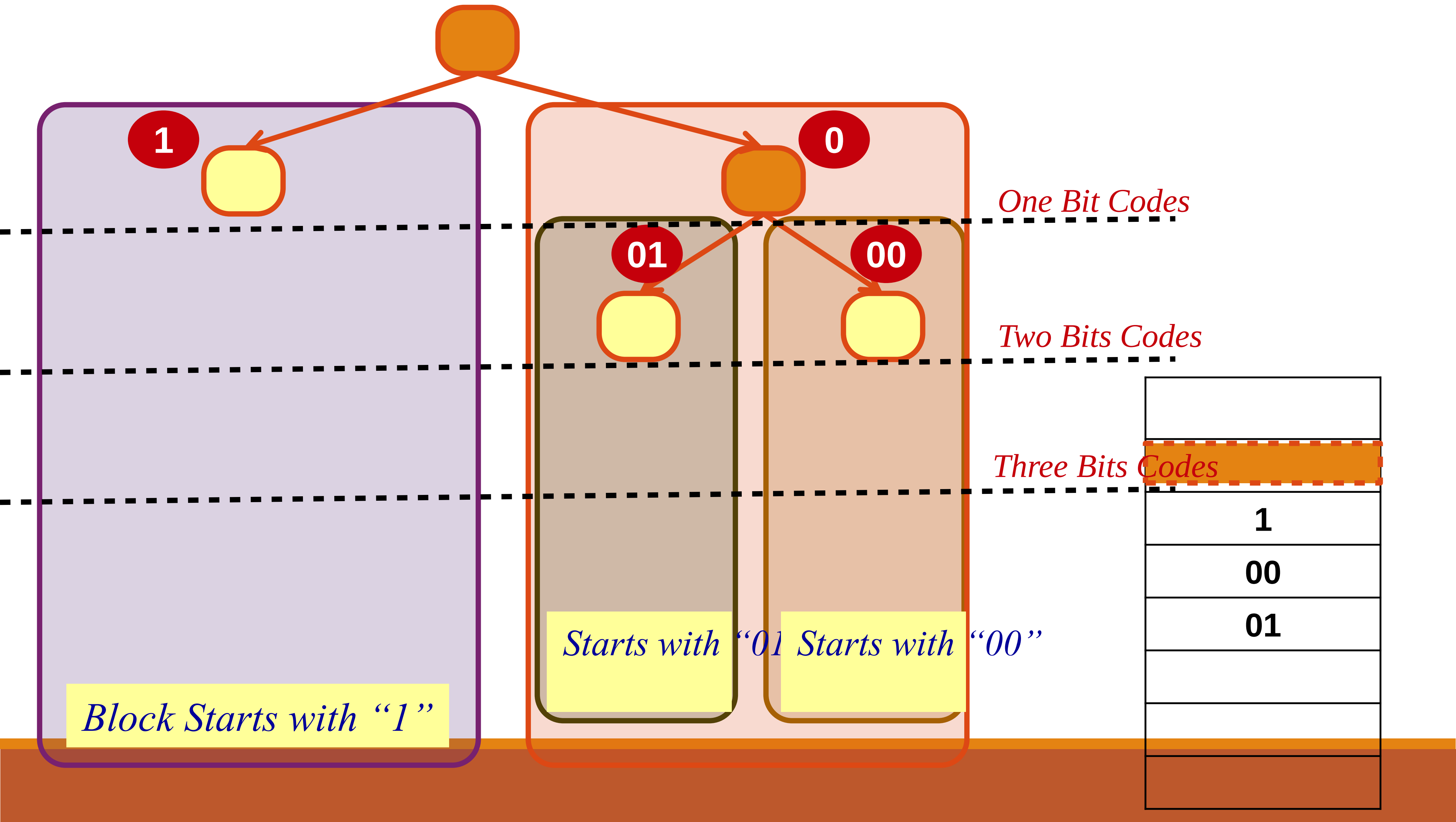
Prefix Conditional Code Generation

No Symbol Code is a prefix of any other Symbol Code

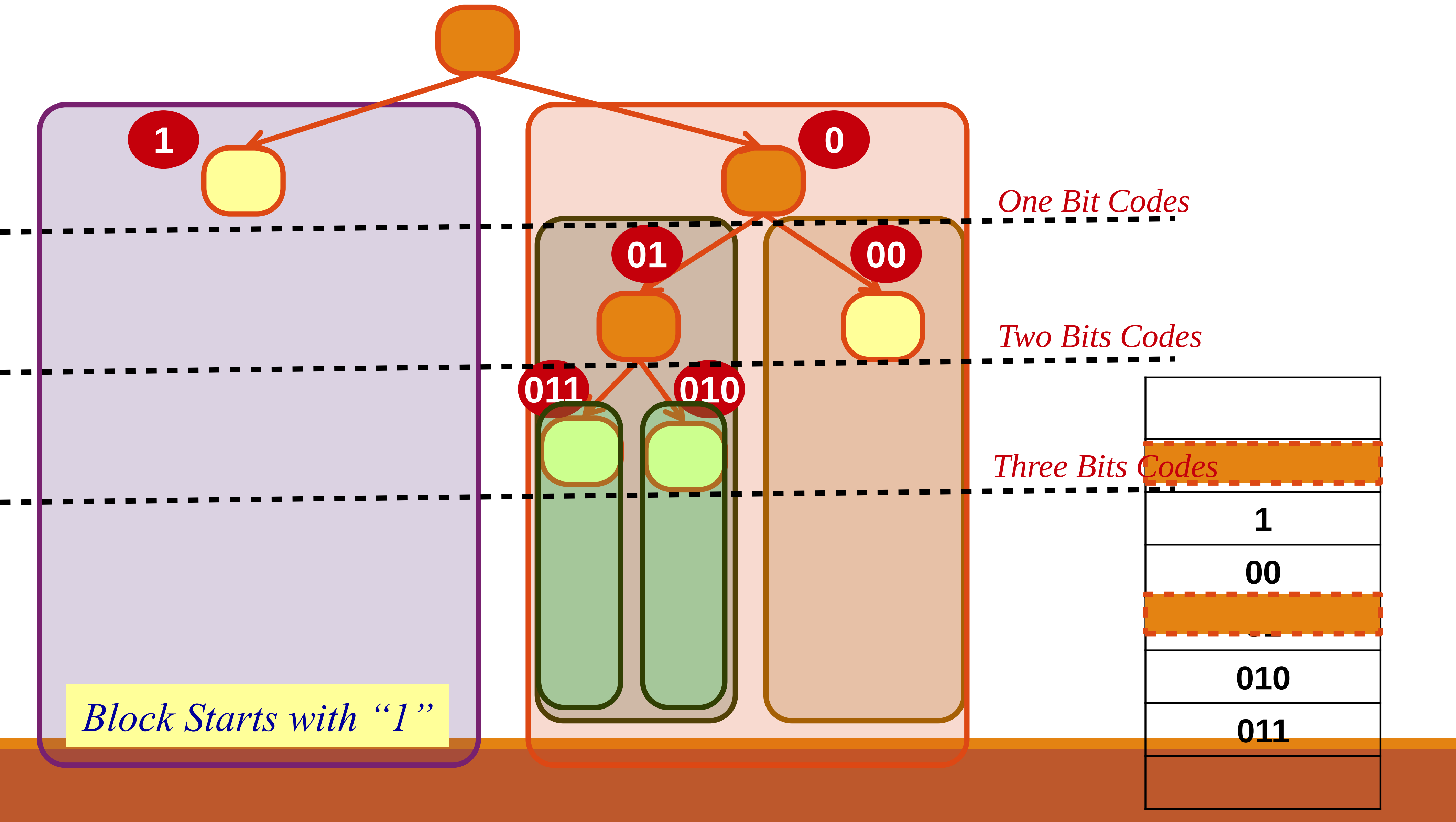
Generation of Prefix Conditional Codes



Generation of Prefix Conditional Codes



Generation of Prefix Conditional Codes



Entropy

Entropy

Entropy is the average amount of information contained in each message received.

Entropy is a measure of *information content:* the number of bits *actually* required to store data.

Entropy of Source Data

Information content "I" associated with any symbol "S" is reversely proportional to its probability

$$I(S) = \mathrm{Log}_2 \{1/P(S)\}$$

Information Content "I" associated with ALL Symbols (S0, S1, S2, ..Sn)=

$$I(\text{All Symbols}) = \mathrm{Log}_2 \{1/P(S_1)\} + \mathrm{Log}_2 \{1/P(S_2)\} + \mathrm{Log}_2 \{1/P(S_3)\} + \ldots + \mathrm{Log}_2 \{1/P(S_n)\}$$

$$I(AllSymbols) = \sum_{i=0}^{i=n} \log_2 \{1/P(S_i)\}$$

Entropy of Source Data

Average Information Content "H" associated with ALL Symbols (S0, S1, S2, ..Sn)=

$$H(S)=1/M\ [m_1 \text{ Log}_2\ \{1/P(S_1)\} + m_2 \text{ Log}_2\ \{1/P(S_2)\} + m_3 \text{ Log}_2\ \{1/P(S_3)\} + ... + m_n \text{ Log}_2\ \{1/P(S_n)\}]$$

Where S_1 is repeated m_1 times, S_2, is repeated m_2 times, , …
and Total Number of Symbols $M= m_1+m_2+..m_n$

$$H(S)=(m_1/M) \text{ Log}_2 \{1/P(S_1)\} + (m_2 /M) \text{ Log}_2\{1/P(S_2)\} + (m_3/M) \text{ Log}_2\{1/P(S_3)\} + ... + (m_n/M) \text{ Log}_2\{1/P(S_n)\}$$

Entropy of Source Data

$$H(S)=(P(S_1)\, \mathrm{Log}_2 \{1/P(S_1)\} + (P(S_2)\, \mathrm{Log}_2\{1/P(S_2)\} + (P(S_3)\, \mathrm{Log}_2\{1/P(S_3)\} + \ldots + (P(S_n)\, \mathrm{Log}_2\{1/P(S_n)\}$$

$$H(S) = \sum_{i=0}^{i=n} P(S_i) \log_2 \{1/P(S_i)\}$$

Entropy of Source Data

$$H(S) = \sum_{i=0}^{i=n} P(S_i) \log_2 \{1/P(S_i)\}$$

Note:

$\log_2 X = \log_{10} X / \log_{10} 2 = \log_{10} X / 0.301$

Calculate Entropy for the following Data **a b a a c a a d a a**

Symbol	P(S)	Log_2 [1/P(S)]	P(S) * Log_2 [1/P(S)]
a	0.7	0.5145	0.36
b	0.1	3.3219	0.33219
c	0.1	3.3219	0.33219
d	0.1	3.3219	0.33219

Entropy H(S) = 1.35687 Bits / symbol

Shannon Source Coding Theorem

For a Discrete Memoryless System (Where no Prediction is Allowed) The maximum level of compression can be reached is the Entropy H(S) measured in Bits/ Symbol

Entropy Calculation Example

abaacaadaa *(10 Symbols)*

P(a)=7/10, P(b)=1/10, P(c)=1/10, P(d)=1/10

$$H(S) = \sum_{i=0}^{i=n} P(S_i) \log_2 \{1/P(S_i)\}$$

Note:

$\log_2 X = \log_{10} X / \log_{10} 2 = \log_{10} X / 0.301$

$H(S) = 0.7 * \log_2 (1/0.7) + 0.1 * \log_2 (1/0.1) + 0.1 * \log_2 (1/0.1) + 0.1 * \log_2 (1/0.1) =$

$0.7 * 0.515 + 0.1 * 3.322 * 3 =$ **1.375 Bits / Symbol**

the Minimum Memory less compression size that can be reached Is 1.375 bits /symbol

(e.g. 10 symbols can be stored in 10*1.375=13.75 bits ~=14 bits

Different Codes for Symbols

abaacaadaa *(10 Symbols)*

P(a)=7/10, P(b)=1/10, P(c)=1/10, P(d)=1/10

Coding 1: Binary System

Symbol	Count	Code	Code Length	Total Bits
a	7	00	2	14
b	1	01	2	2
c	1	10	2	2
d	1	11	2	2

Compressed Total = 20 Bits

Coding 2: Huffman

Symbol	Count	Code	Code Length	Total Bits
a	7	0	1	7
b	1	10	2	2
c	1	110	3	3
d	1	111	3	3

Compressed Total = 15 Bits

Compressed (According to Entropy) = 14 Bits

Entropy Calculation Example

P(a)=0.17, P(b)=0.22, P(c)=0.15, P(d)=0.14, P(e)=0.3, P(f)=0.02

$$H(S) = \sum_{i=0}^{i=n} P(S_i) \log_2 \{1/P(S_i)\}$$

Note:

$\log_2 X = \log_{10} X / \log_{10} 2$

$H(S) = 0.17 * \log_2 (1/0.17) + 0.22 * \log_2 (1/0.22) + 0.15 * \log_2 (1/0.15) + 0.14 * \log_2 (1/0.14) + 0.30 * \log_2 (1/0.30) + 0.02 * \log_2 (1/0.02) =$ **2.3567 Bits / Symbol**

the Minimum compression size that can be reached is **2.3567 bits /symbol**

(e.g. 100 symbols can be stored in 100*2.3567=235.67 bits ~=236 bits

Huffman Coding Algorithm

Huffman Coding Algorithm

- Each symbol is a leave node in a tree
- Combining the two symbols or composite symbols with the least probabilities to form a new parent composite symbols, which has the combined probabilities. Assign a bit 0 and 1 to the two links
- Continue this process till all symbols merged into one root node. For each symbol, the sequence of the 0s and 1s from the root node to the symbol is the code

Huffman Coding Example

P(a)=0.17, P(b)=0.22, P(c)=0.15, P(d)=0.14, P(e)=0.3, P(f)=0.02

P(e)= 0.3

P(b)= 0.22

P(a)= 0.17

P(c)= 0.15

P(d)= 0.14

P(f)= 0.02

Order Symbols According to Their probabilities (Descending)

Huffman Coding Example

P(e)= 0.3	P(e)= 0.3
P(b)= 0.22	P(b)= 0.22
P(a)= 0.17	P(a)= 0.17
P(c)= 0.15	P(d+f)= 0.16
P(d)= 0.14	P(c)= 0.15
P(f)= 0.02	

Combine Last two Symbols (with Lowest Probabilities),
Then reorder the list

Huffman Coding Example

P(e)= 0.3	P(e)= 0.3	P(d+f+c)= 0.31
P(b)= 0.22	P(b)= 0.22	P(e)= 0.3
P(a)= 0.17	P(a)= 0.17	P(b)= 0.22
P(c)= 0.15	P(d+f)= 0.16	P(a)= 0.17
P(d)= 0.14	P(c)= 0.15	
P(f)= 0.02		

Combine Last two Symbols (with Lowest Probabilities),
Then reorder the list

Huffman Coding Example

P(e)= 0.3	P(e)= 0.3	P(d+f+c)= 0.31	P(b+a)= 0.39
P(b)= 0.22	P(b)= 0.22	P(e)= 0.3	P(d+f+c)=0.31
P(a)= 0.17	P(a)= 0.17	P(b)= 0.22	P(e)= 0.30
P(c)= 0.15	P(d+f)= 0.16	P(a)= 0.17	
P(d)= 0.14	P(c)= 0.15		
P(f)= 0.02			

Combine Last two Symbols (with Lowest Probabilities),
Then reorder the list

Huffman Coding Example

P(e)= 0.3	P(e)= 0.3	P(d+f+c)= 0.31	P(b+a)= 0.39	P(d+f+c+e)= 0.61
P(b)= 0.22	P(b)= 0.22	P(e)= 0.3	P(d+f+c)=0.31	P(b+a)=0.39
P(a)= 0.17	P(a)= 0.17	P(b)= 0.22	P(e)= 0.30	
P(c)= 0.15	P(d+f)= 0.16	P(a)= 0.17		
P(d)= 0.14	P(c)= 0.15			
P(f)= 0.02				

Combine Last two Symbols (with Lowest Probabilities),
Then reorder the list

Huffman Coding Example

P(e)= 0.3
P(b)= 0.22
P(a)= 0.17
P(c)= 0.15
P(d)= 0.14
P(f)= 0.02

P(e)= 0.3
P(b)= 0.22
P(a)= 0.17
P(d+f)= 0.16
P(c)= 0.15

P(d+f+c)= 0.31
P(e)= 0.3
P(b)= 0.22
P(a)= 0.17

P(b+a)= 0.39
P(d+f+c)=0.31
P(e)= 0.30

0
P(d+f+c+e)= 0.61
1
P(b+a)=0.39

Symbol	Code
a	
b	
c	
d	
e	
f	

Assign Binary Codes to each branch, Continue

Huffman Coding Example

P(e)= 0.3
P(b)= 0.22
P(a)= 0.17
P(c)= 0.15
P(d)= 0.14
P(f)= 0.02

P(e)= 0.3
P(b)= 0.22
P(a)= 0.17
P(d+f)= 0.16
P(c)= 0.15

P(d+f+c)= 0.31
P(e)= 0.3
P(b)= 0.22
P(a)= 0.17

1
P(b+a)= 0.39
00
P(d+f+c)=0.31
P(e)= 0.30
01

0
P(d+f+c+e)= 0.61
1
P(b+a)=0.39

Symbol	Code
a	
b	
c	
d	
e	01
f	

Assign Binary Codes to each branch, Continue

Huffman Coding Example

P(e)= 0.3
P(b)= 0.22
P(a)= 0.17
P(c)= 0.15
P(d)= 0.14
P(f)= 0.02

P(e)= 0.3
P(b)= 0.22
P(a)= 0.17
P(d+f)= 0.16
P(c)= 0.15

P(d+f+c)= 0.31
P(e)= 0.3
P(b)= 0.22
P(a)= 0.17

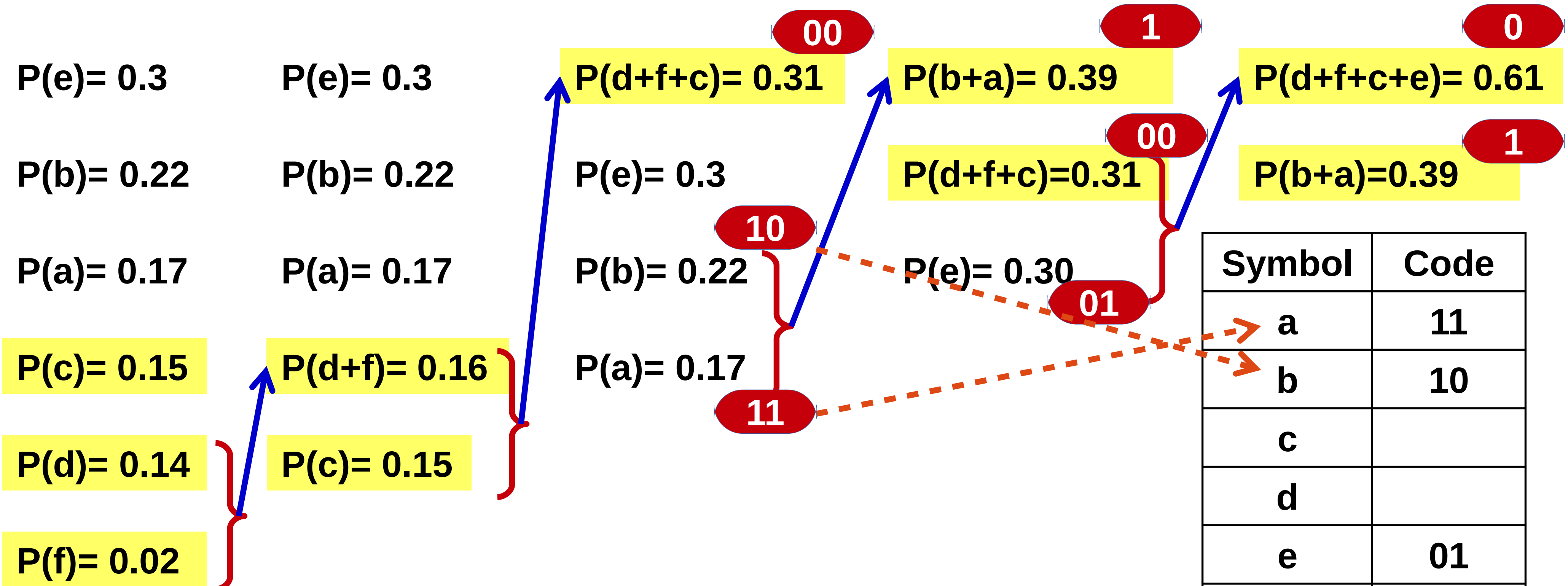
1
P(b+a)= 0.39
00
P(d+f+c)=0.31
P(e)= 0.30
01

0
P(d+f+c+e)= 0.61
1
P(b+a)=0.39

Symbol	Code
a	
b	
c	
d	
e	01
f	

Assign Binary Codes to each branch, Continue

Huffman Coding Example



Symbol	Code
a	11
b	10
c	
d	
e	01
f	

Assign Binary Codes to each branch, Continue

Huffman Coding Example

- P(e)= 0.3
- P(b)= 0.22
- P(a)= 0.17
- P(c)= 0.15
- P(d)= 0.14
- P(f)= 0.02

- P(e)= 0.3
- P(b)= 0.22
- P(a)= 0.17
- P(d+f)= 0.16 — 000
- P(c)= 0.15 — 001

- P(d+f+c)= 0.31 — 00
- P(e)= 0.3
- P(b)= 0.22 — 10
- P(a)= 0.17 — 11

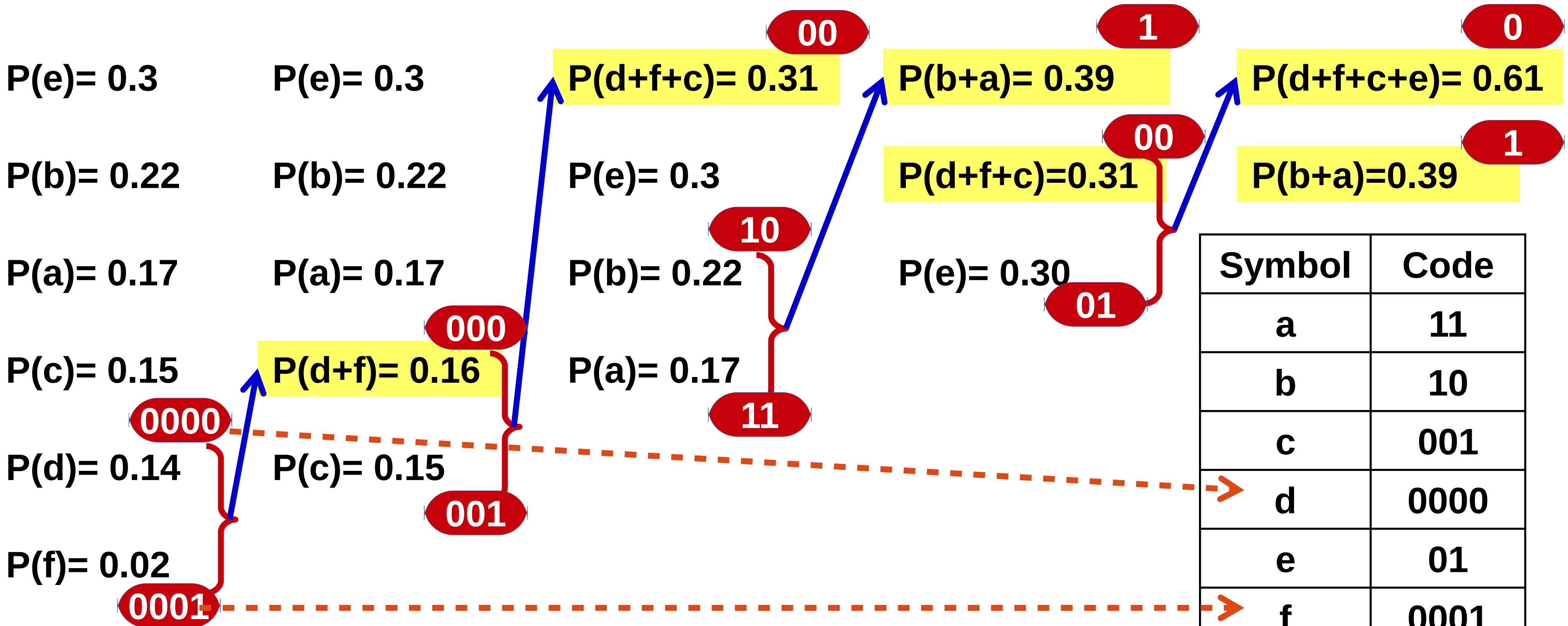
- P(b+a)= 0.39 — 1
- P(d+f+c)=0.31 — 00
- P(e)= 0.30 — 01

- P(d+f+c+e)= 0.61 — 0
- P(b+a)=0.39 — 1

Symbol	Code
a	11
b	10
c	001
d	
e	01
f	

Assign Binary Codes to each branch, Continue

Huffman Coding Example



Symbol	Code
a	11
b	10
c	001
d	0000
e	01
f	0001

Assign Binary Codes to each branch, Continue

Compression Ratio

a	17	11	2	34	
b	22	10	2	44	
c	15	001	3	45	
d	14	0000	4	56	
e	30	01	2	60	
f	2	0001	4	8	

Compressed Total = 247 Bits

Uncompressed size=100 Symbol * 3 bits/symbol = 300 Bits

Entropy =2.35 bits/Symbol (for 100 Symbols H=235 bits

Modified Huffman Coding Algorithm

Modified Huffman Coding Algorithm

- Same steps and concept as Huffman Coding Algorithm
- In order to Minimize Huffman Table size and Codes lengths, set **Minimum Limit of Symbol Probabilities** (e.g. 0.05)
- Symbols with Probabilities <= the Limit will be grouped in one group called "**Others**"
- All Symbols will be coded using Corresponding Huffman codes, Symbols in "Other" group will be coded using both **"Others" Huffman code** + **Original** Symbol Code.

Huffman Coding Example

a b c a z d a f c q d a d c u a b a p d

Count (a) = 6	**P(a)= 0.3**
Count (b) = 2	**P(b)= 0.1**
Count (c) = 3	**P(c)= 0.15**
Count (d) = 4	**P(d)= 0.2**
Count (f) = 1	**P(f)= 0.05**
Count (z) = 1	**P(z)= 0.05**
Count (q) = 1	**P(q)= 0.05**
Count (p) = 1	**P(p)= 0.05**
Count (u) = 1	**P(u)= 0.05**

P(a)= 0.3

P(b)= 0.1

P(c)= 0.15

P(d)= 0.2

P(Others)= 0.25

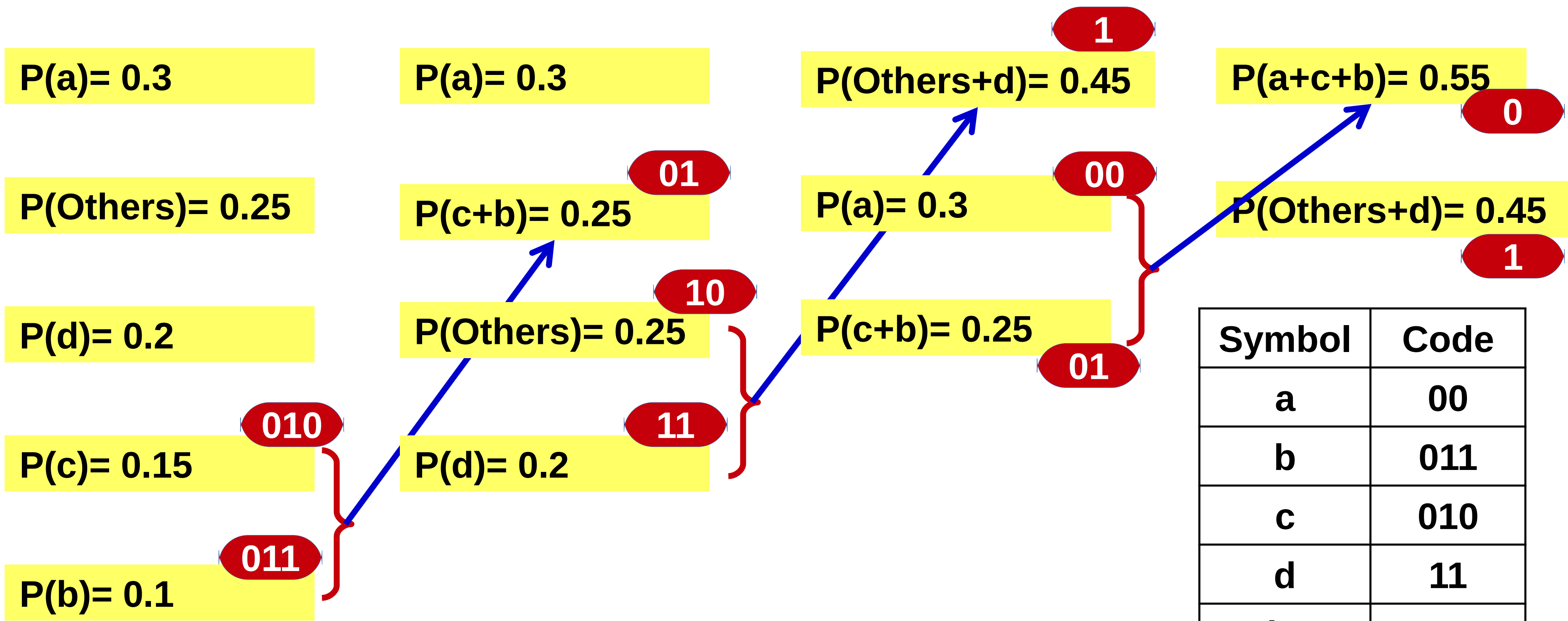
Symbol	Original Code
a	0000
b	0001
c	0010
d	0011
f	0100
p	0101
q	0110
u	0111
z	1000

Modified Huffman Coding Example

P(a)= 0.3	P(a)= 0.3	P(Others+d)= 0.45	P(a+c+b)= 0.55
P(Others)= 0.25	P(c+b)= 0.25	P(a)= 0.3	P(Others+d)= 0.45
P(d)= 0.2	P(Others)= 0.25	P(c+b)= 0.25	
P(c)= 0.15	P(d)= 0.2		
P(b)= 0.1			

Combine Last two Symbols (with Lowest Probabilities),
Then reorder the list

Modified Huffman Coding Example



Symbol	Code
a	00
b	011
c	010
d	11
Others	10

Assign Binary Codes to each branch, Continue

Modified Huffman Coding Example

a b c a z d a f c q d a d c u a b a p d

00 011 010 00 101000 11 00 100100 010 100110 11 00 11 010 …….

A	b	c	d	Others
6*2 +	2*3 +	3*3 +	4*2 +	5 (2+4) =

12 + 6 + 9 + 8 + 30 = 65 bits

Symbol	Code
a	00
b	011
c	010
d	11
Others	10

Symbol	Original Code
a	0000
b	0001
c	0010
d	0011
f	0100
p	0101
q	0110
u	0111
z	1000

Example 2: Huffman Code Construction

In the following Table

Given ***Data Symbols (Character)*** and

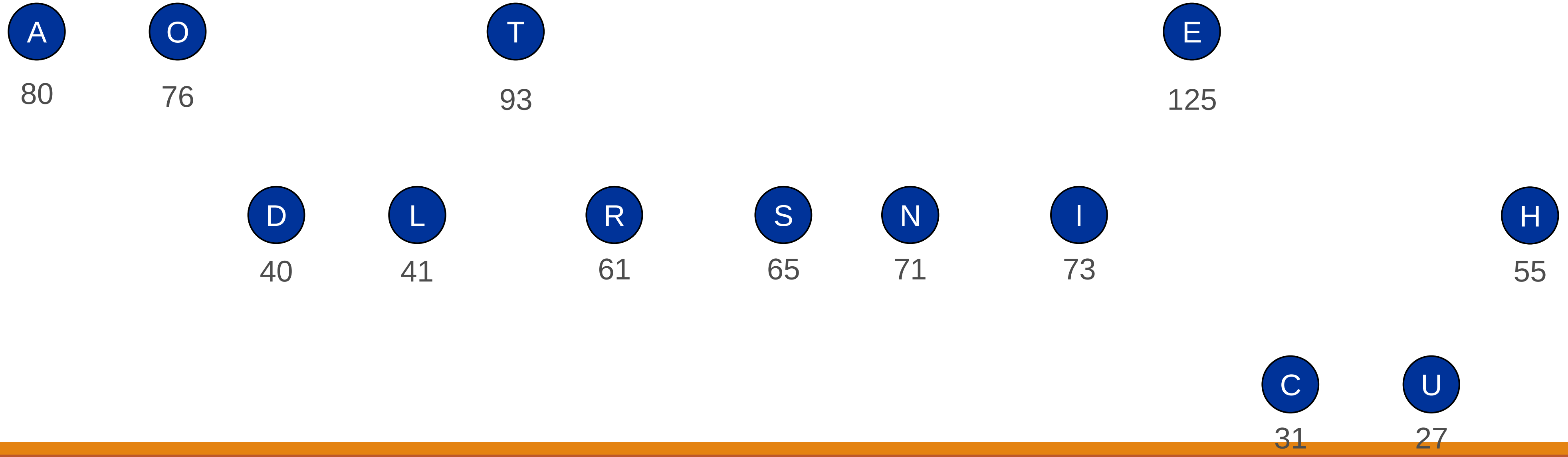
Corresponding ***Number of Occurrence (Frequency)***

Use Standard Huffman Coding to

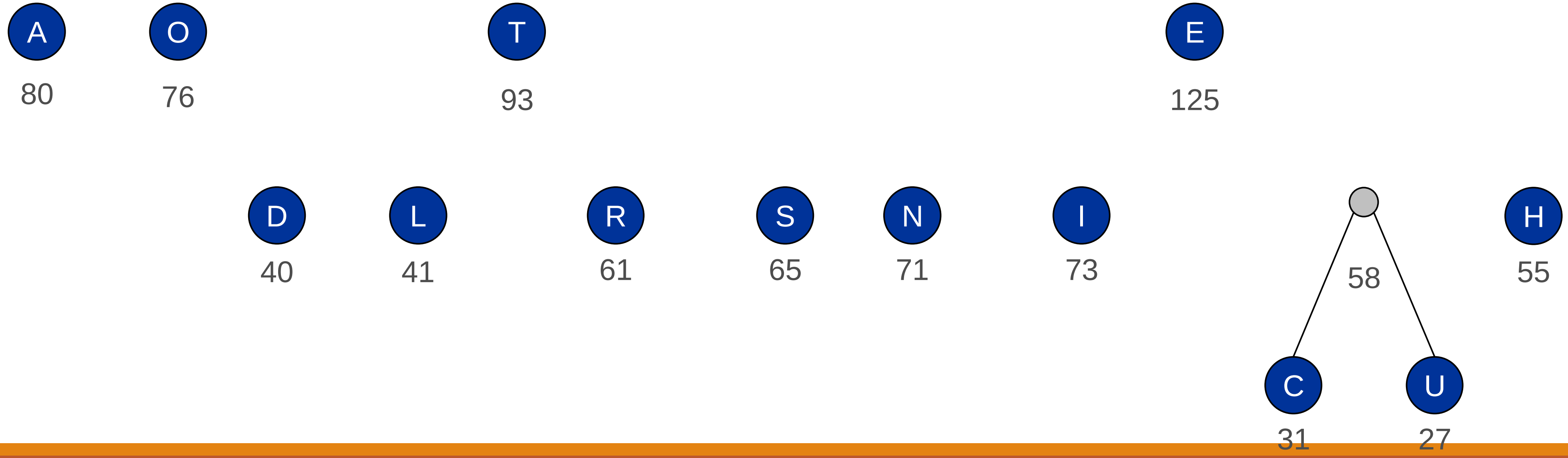
generate VLC for each Symbol

Char	Freq
E	125
T	93
A	80
O	76
I	72
N	71
S	65
R	61
H	55
L	41
D	40
C	31
U	27

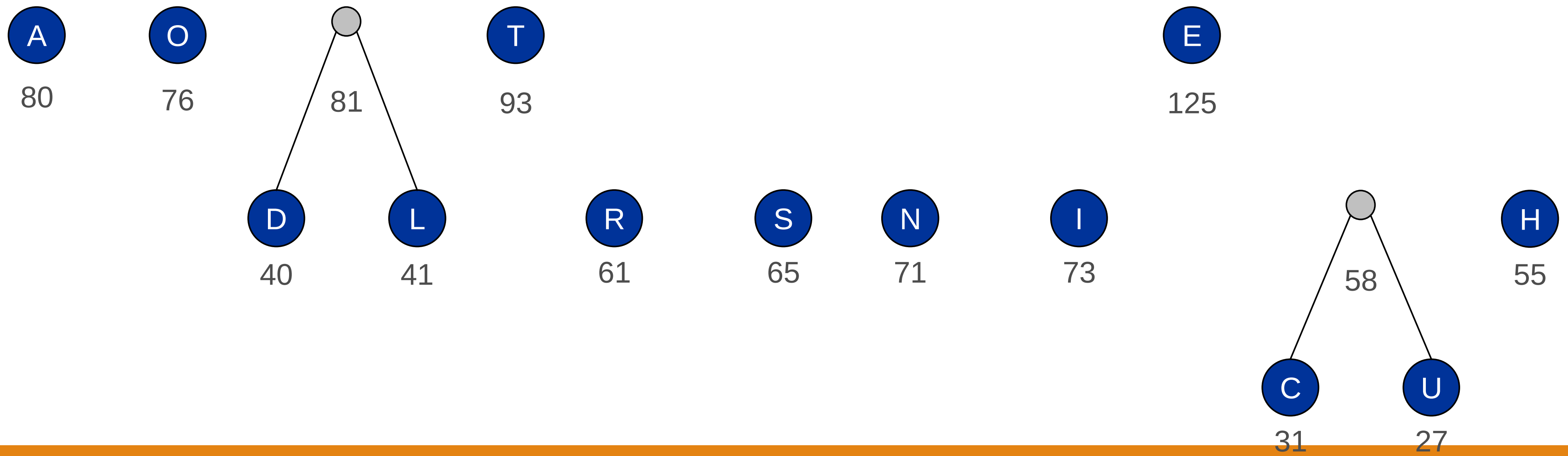
Example 2: Huffman Code Construction



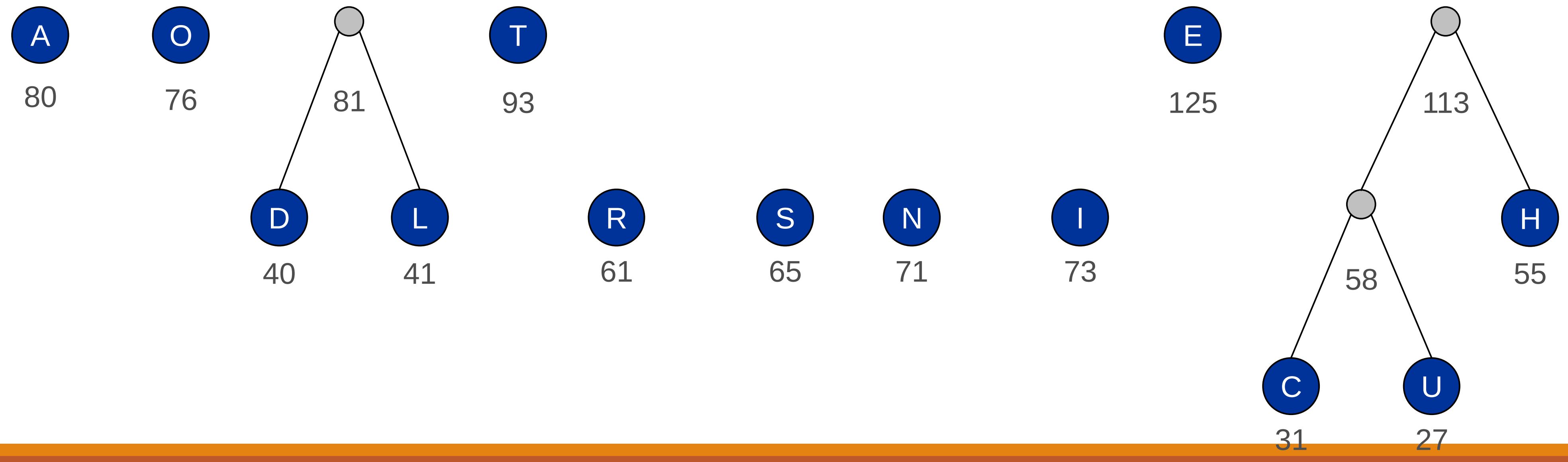
Example2: Huffman Code Construction



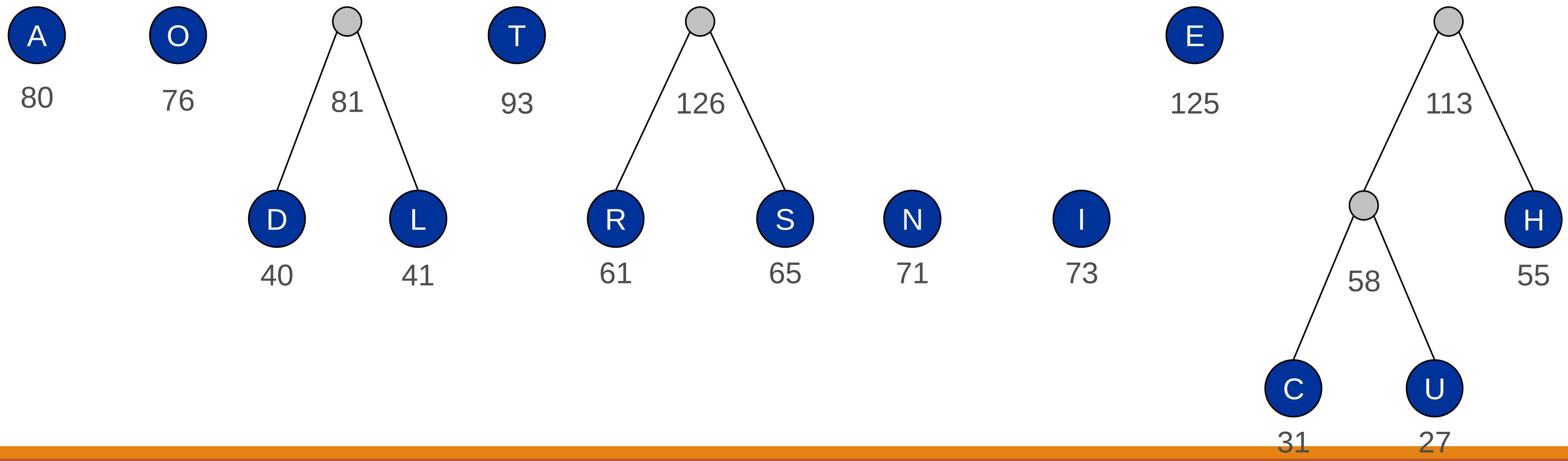
Example2: Huffman Code Construction



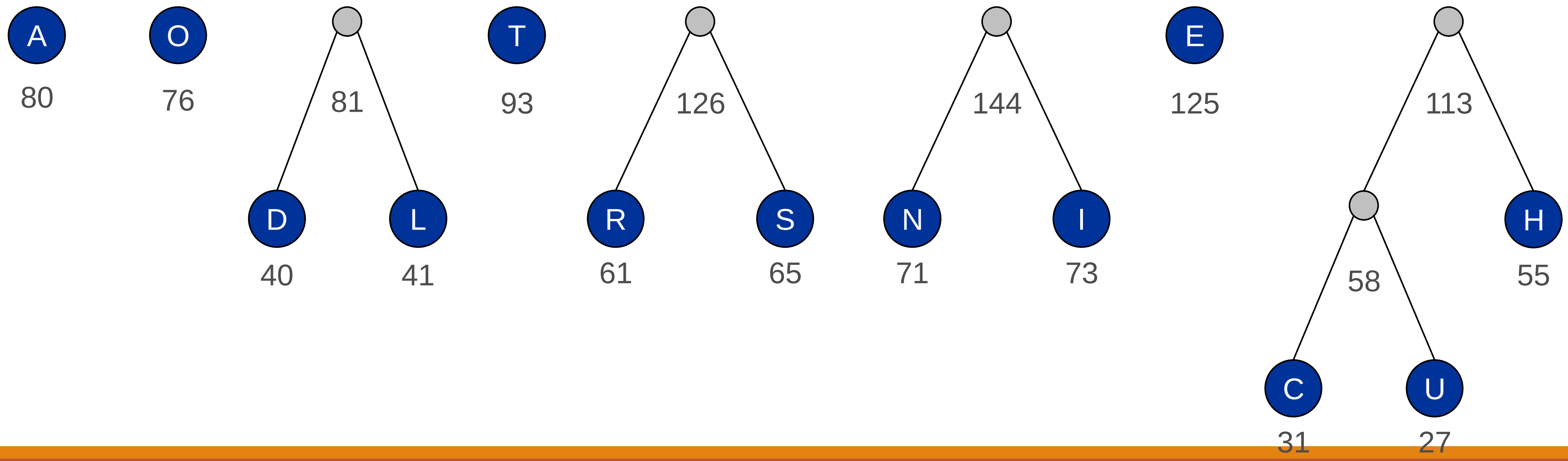
Example2: Huffman Code Construction



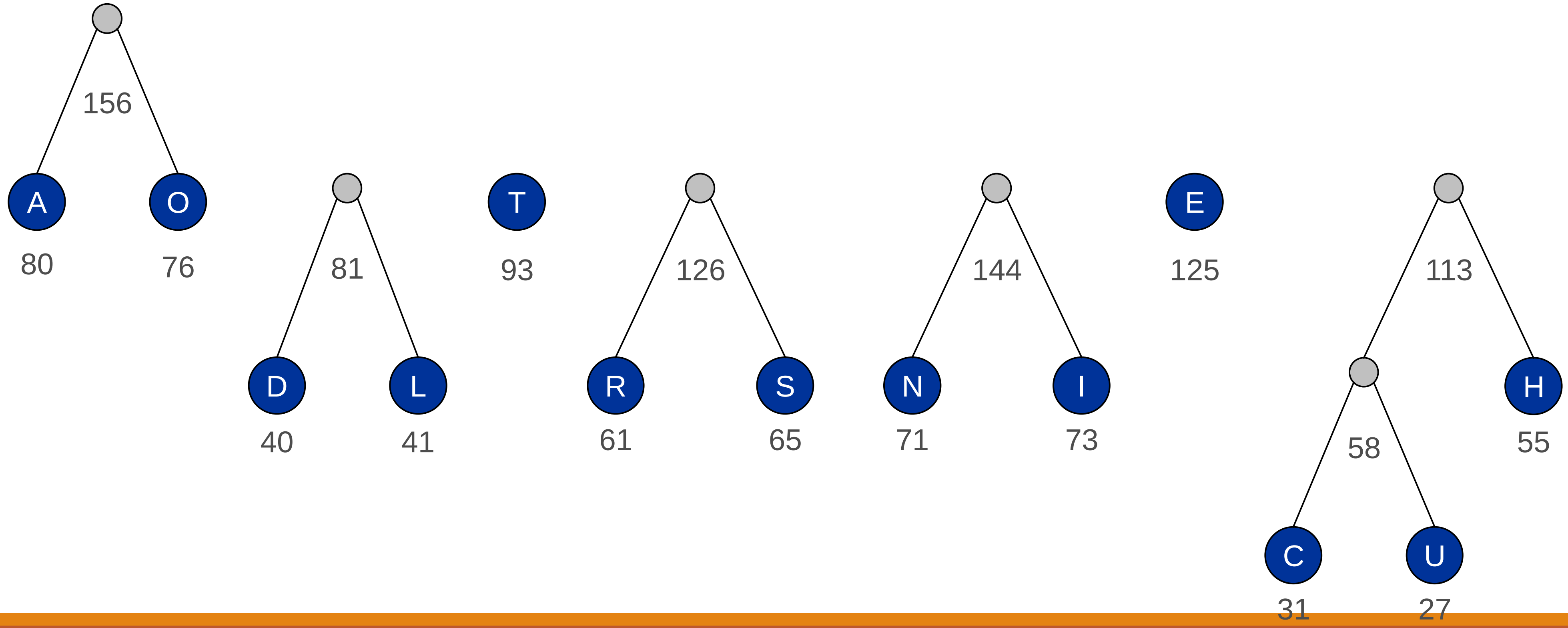
Example2: Huffman Code Construction



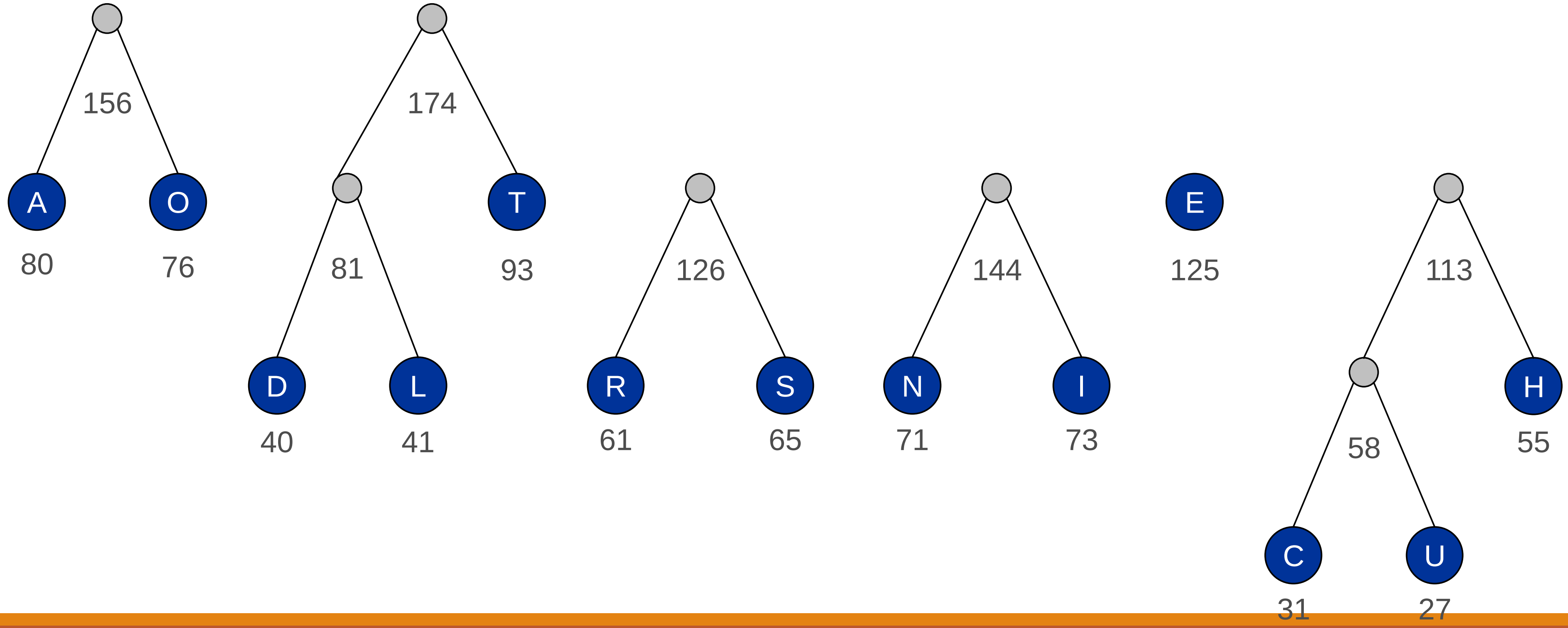
Example2: Huffman Code Construction



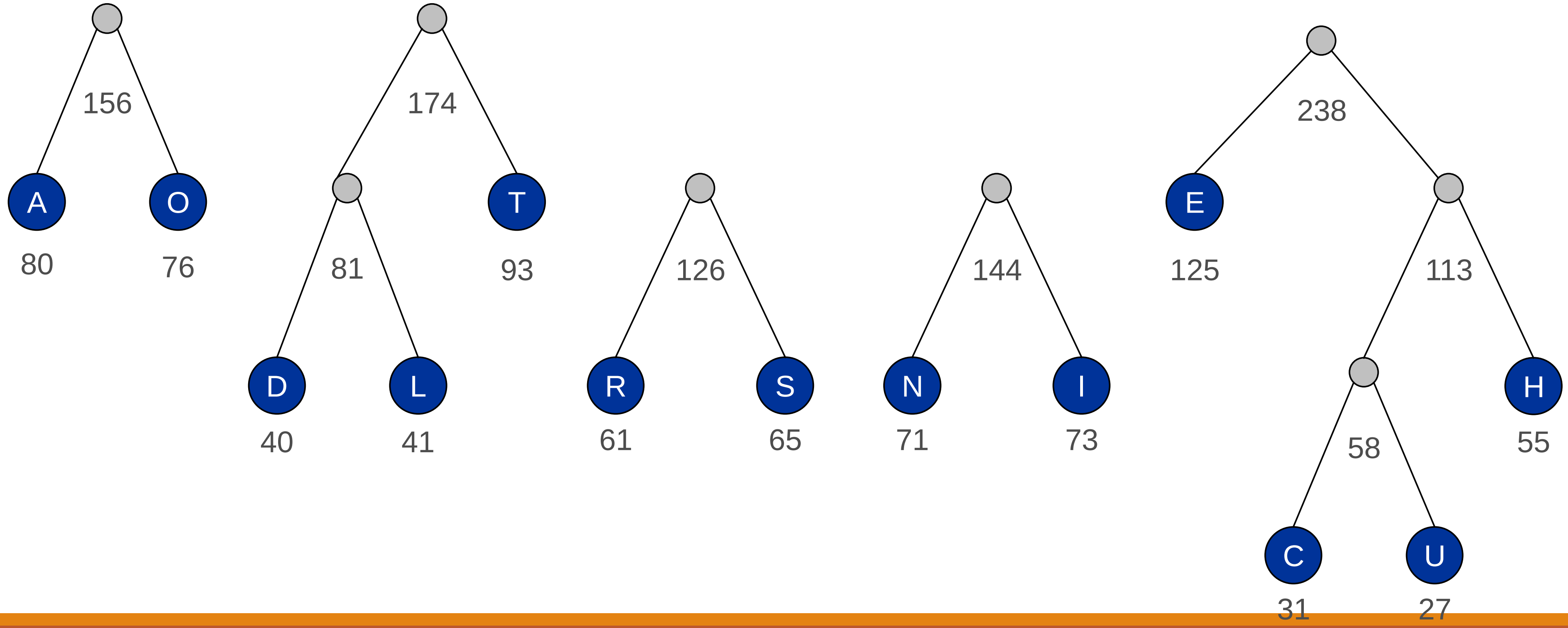
Example2: Huffman Code Construction



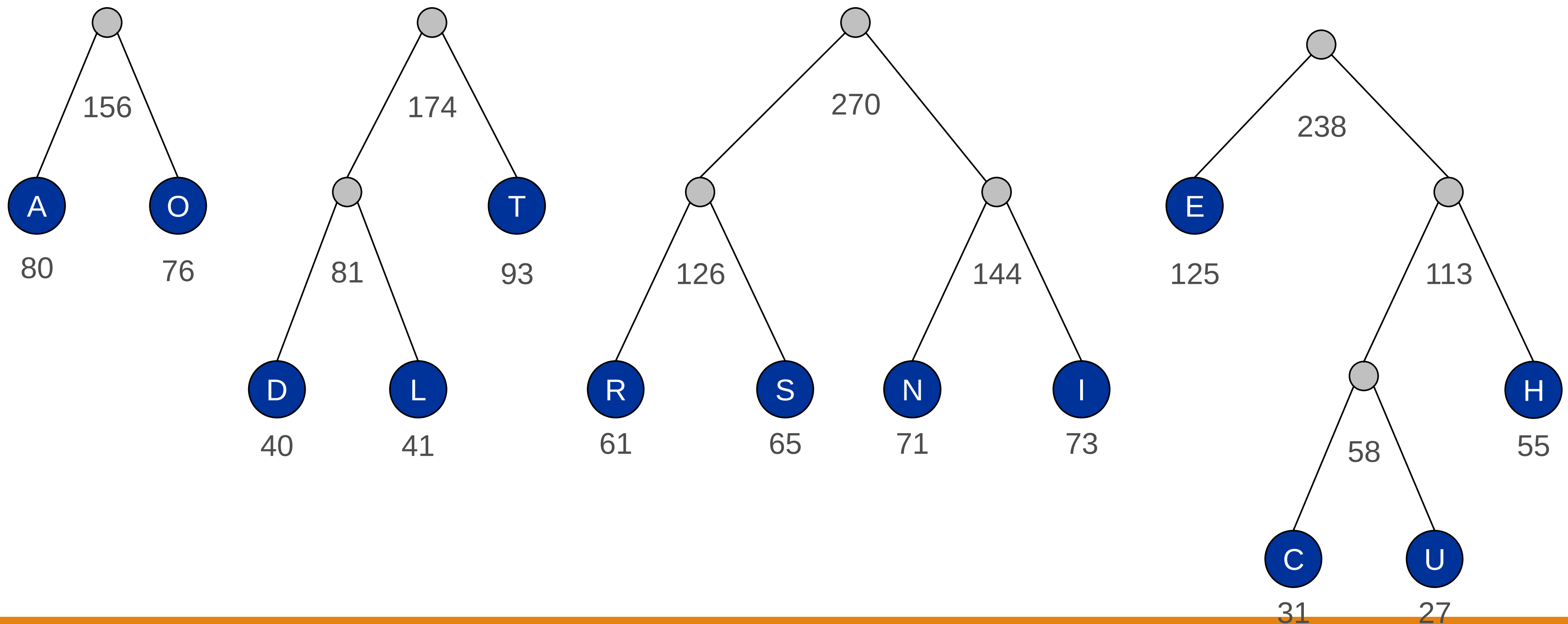
Example2: Huffman Code Construction



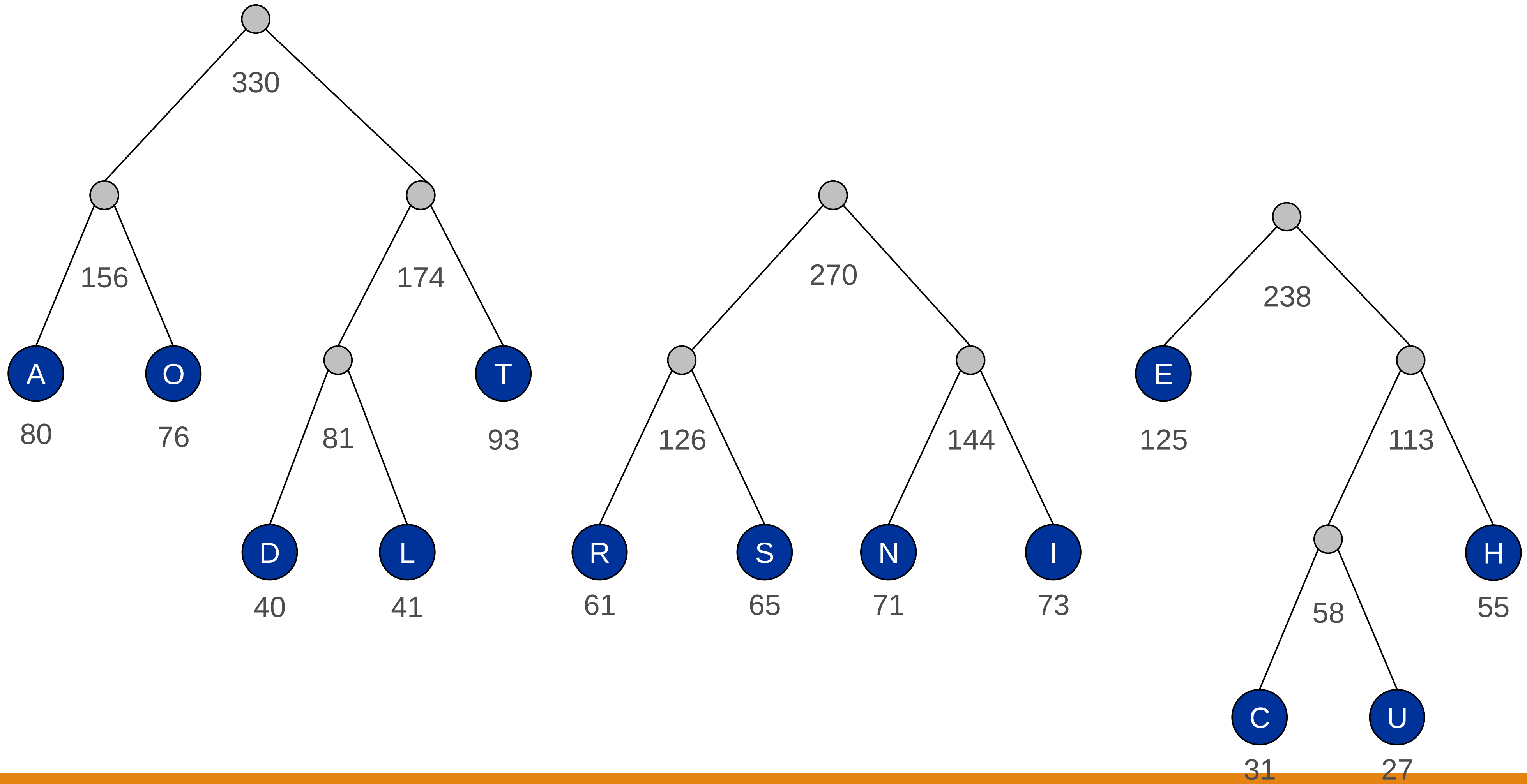
Example2: Huffman Code Construction



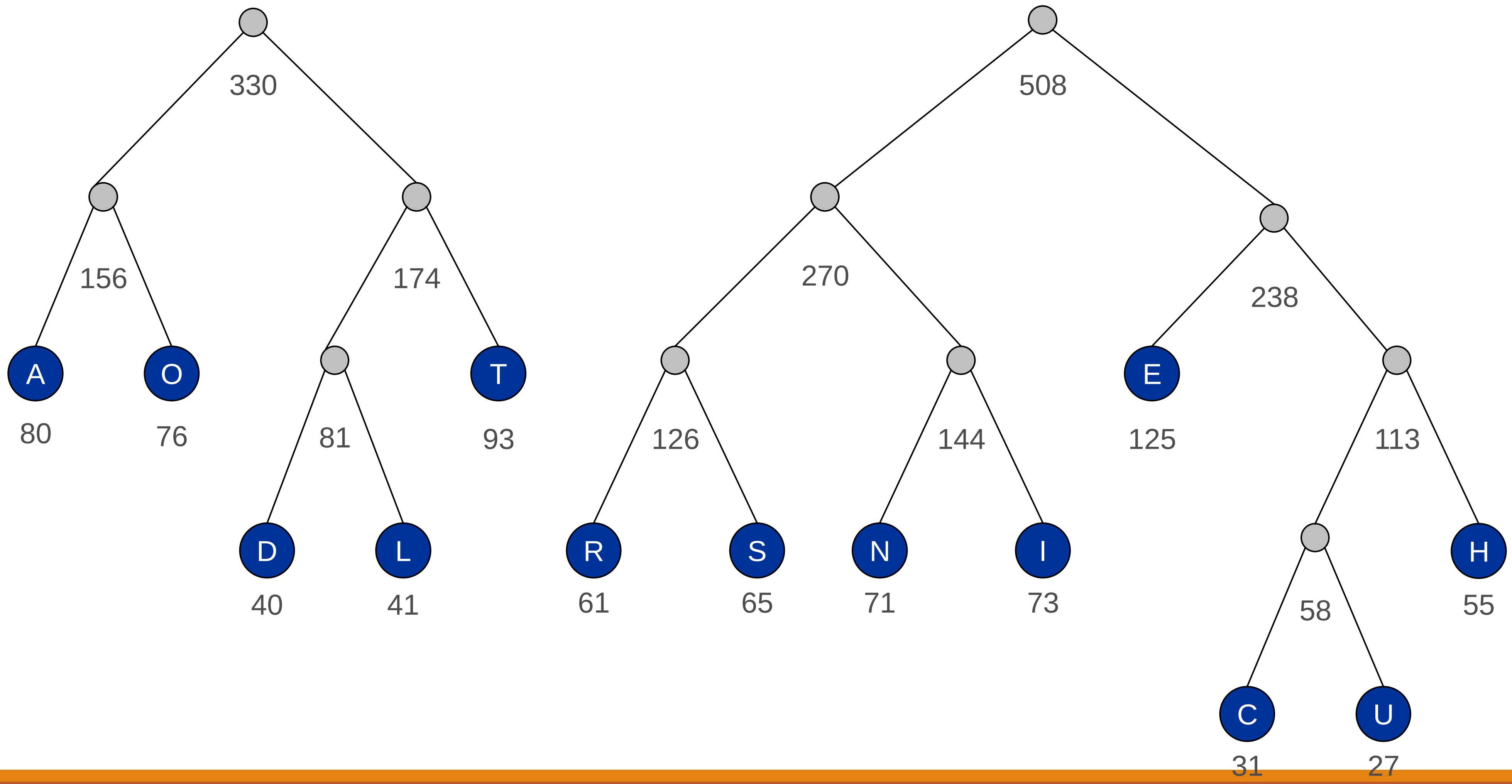
Example2: Huffman Code Construction



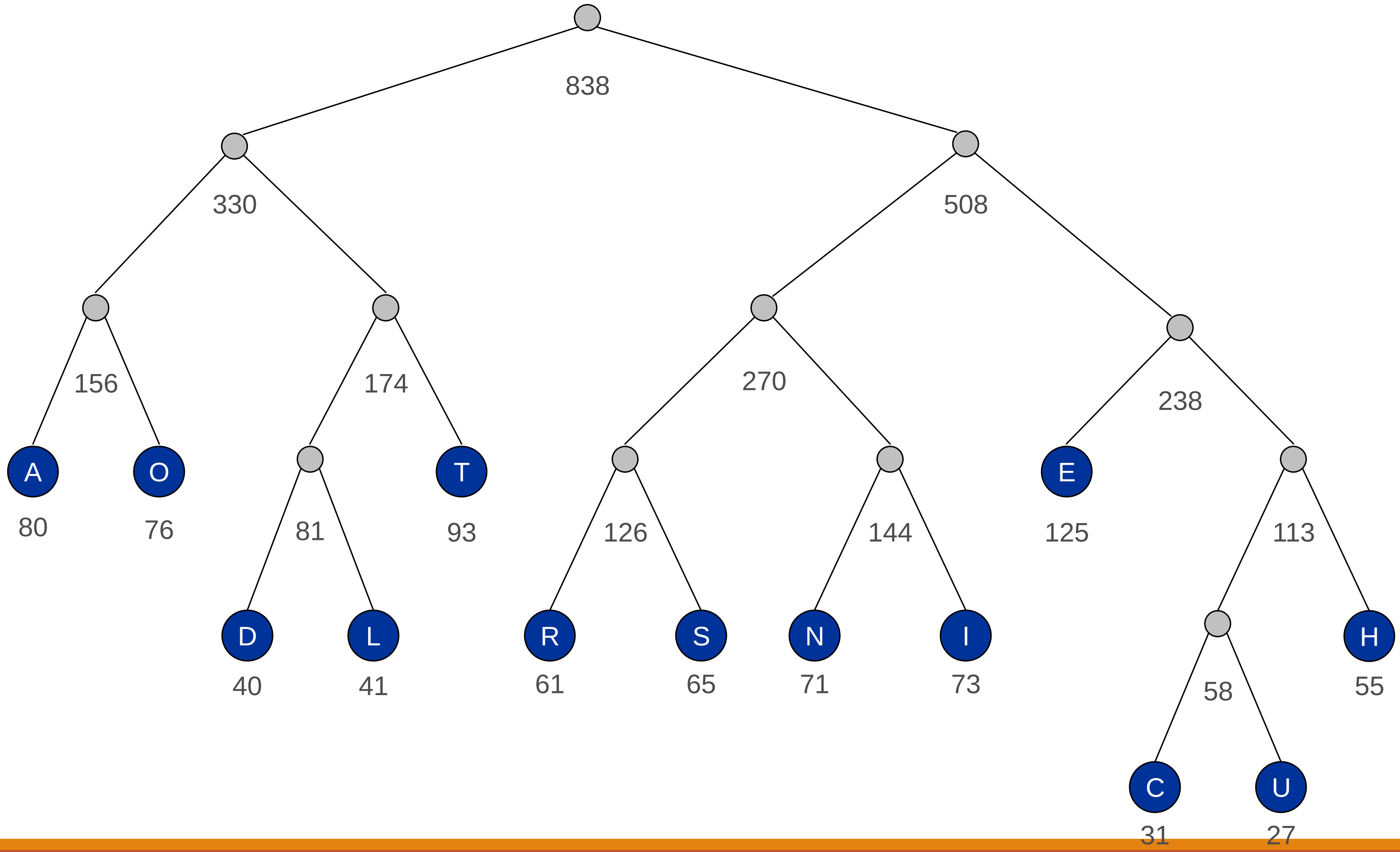
Example2: Huffman Code Construction



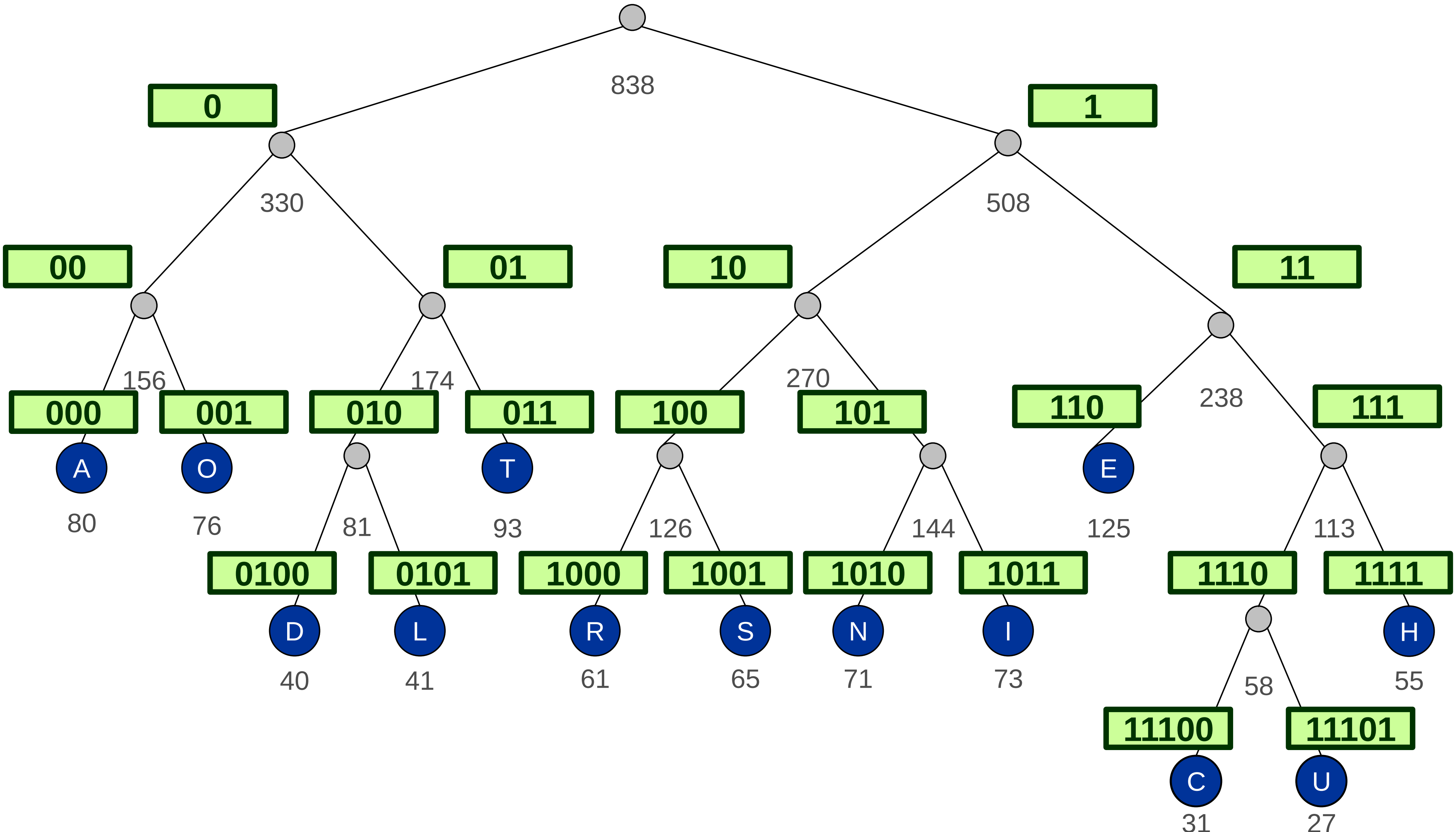
Example2: Huffman Code Construction



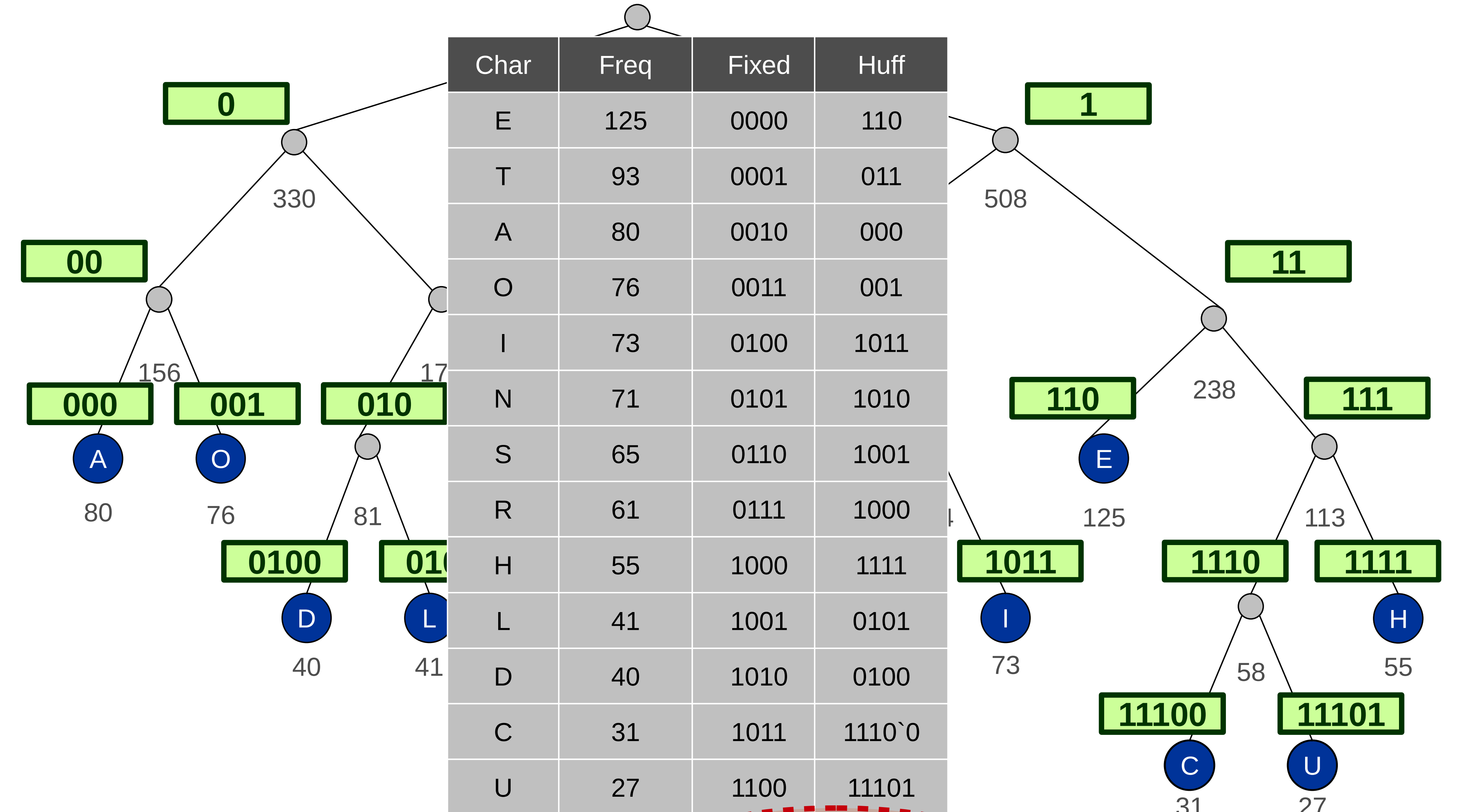
Example2: Huffman Code Construction



Example Huffman Code Construction



Example Huffman Code Construction



Char	Freq	Fixed	Huff
E	125	0000	110
T	93	0001	011
A	80	0010	000
O	76	0011	001
I	73	0100	1011
N	71	0101	1010
S	65	0110	1001
R	61	0111	1000
H	55	1000	1111
L	41	1001	0101
D	40	1010	0100
C	31	1011	1110`0
U	27	1100	11101
Total	838	4.00	3.62

Bits/Symbol