Huffman Code

- Data Compression
  - Lossy and Lossless Compression
  - Adaptive and non-Adaptive Compression
  - Fixed and Variable length Codes
- Prefix Free Codes
  - Binary Tree Representation
  - Goodness Measure
- Generic Greedy Algorithm
- Huffman Code
- Optimality and Implementation

Imdad ullah Khan

Fixed versus Variable length codes

Fixed Length Code

- Fixed number of bits for each symbol (character)
- e.g. ASCII (7 bits) and Unicode (UTF-8, UTF-16)
- ASCII can represent $2^7 = 128$ symbols

Variable Length Code

- Variable number of bits for each symbol
- Can use fewer bits for more frequent symbols
- e.g. Huffman code
- Difficult to find, needs compression scheme

Fixed versus Variable length codes

Characters	a	b	c	d
Fixed-Length Code	00	01	10	11
Variable Length Code 1	0	10	110	111
Variable Length Code 2	0	1	01	10

Let the string be **b a a d a b**

- Fixed Code: 01 00 00 11 00 01 $\rightarrow$ 12 bits
- Variable Code 1: 10 0 0 111 0 10 $\rightarrow$ 10 bits
- Variable Code 2: 1 0 0 10 0 1 $\rightarrow$ 4 bits

Variable Code 2 compresses a lot

Codes must be **uniquely** decodable

Variable Code 2: 1001001 can be decoded as dabac or bacad or ...

Prefix Free Codes

Prefix free: When no code is a prefix of another

If a code is prefix free, then it is uniquely decodable

Characters	a	b	c	d
Fixed-Length Code	00	01	10	11
Variable Length Code 1	0	10	110	111
Variable Length Code 2	0	1	01	10

Variable Length Code 2 is not prefix free

Code for '*a*' (0) is a prefix of code for '*c*' (01)

Visualizing Codes as Binary Trees

Prefix free code can be represented by a rooted binary tree

Leaves are labeled with characters and edges with bits

The bits along the path from root a leaf is code of the symbol

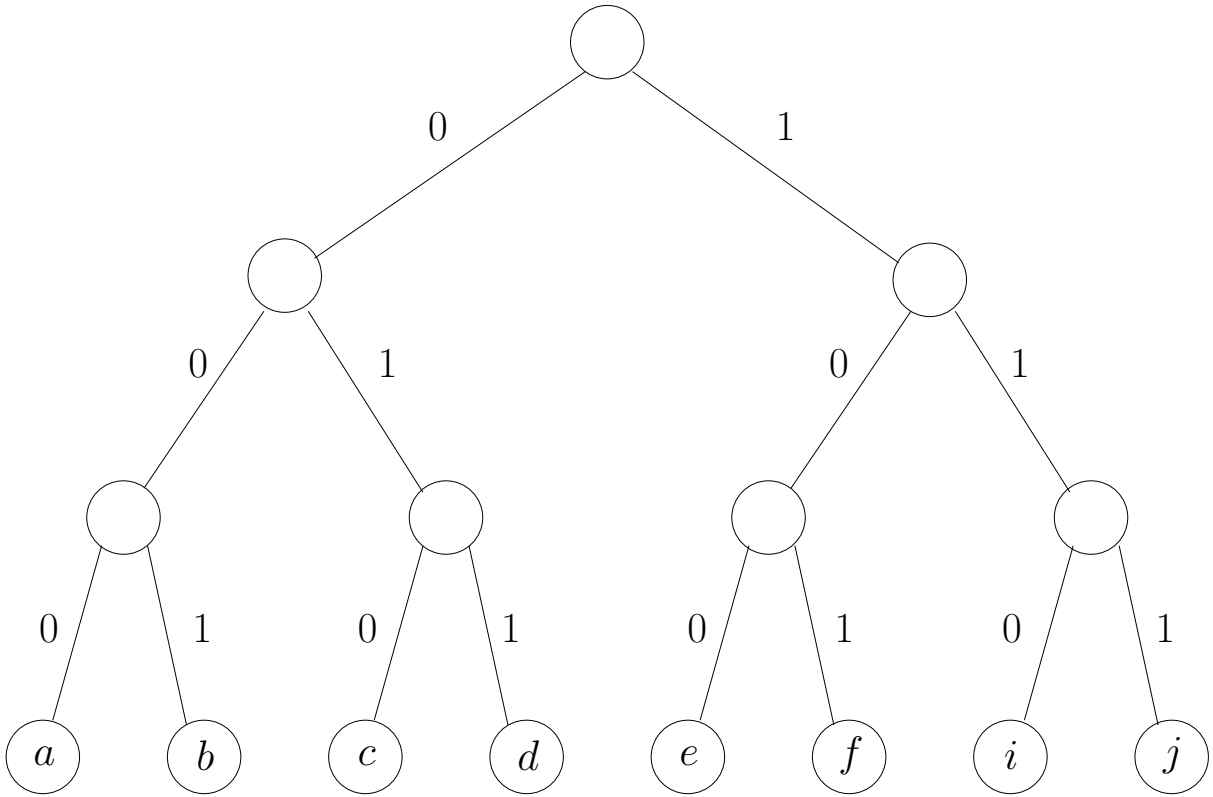
Prefix free condition is equivalent to have symbols at leaves only

Decoding: Read the coded string and start traversing the path from root to a leaf taking left or right child if the bit is 0 or 1

Write the leaf and continue reading next bit and start traversal at root

Fixed Length Codes as Binary Tree

Symbol	Code
a	000
b	001
c	010
d	011
e	100
f	101
i	110
j	111

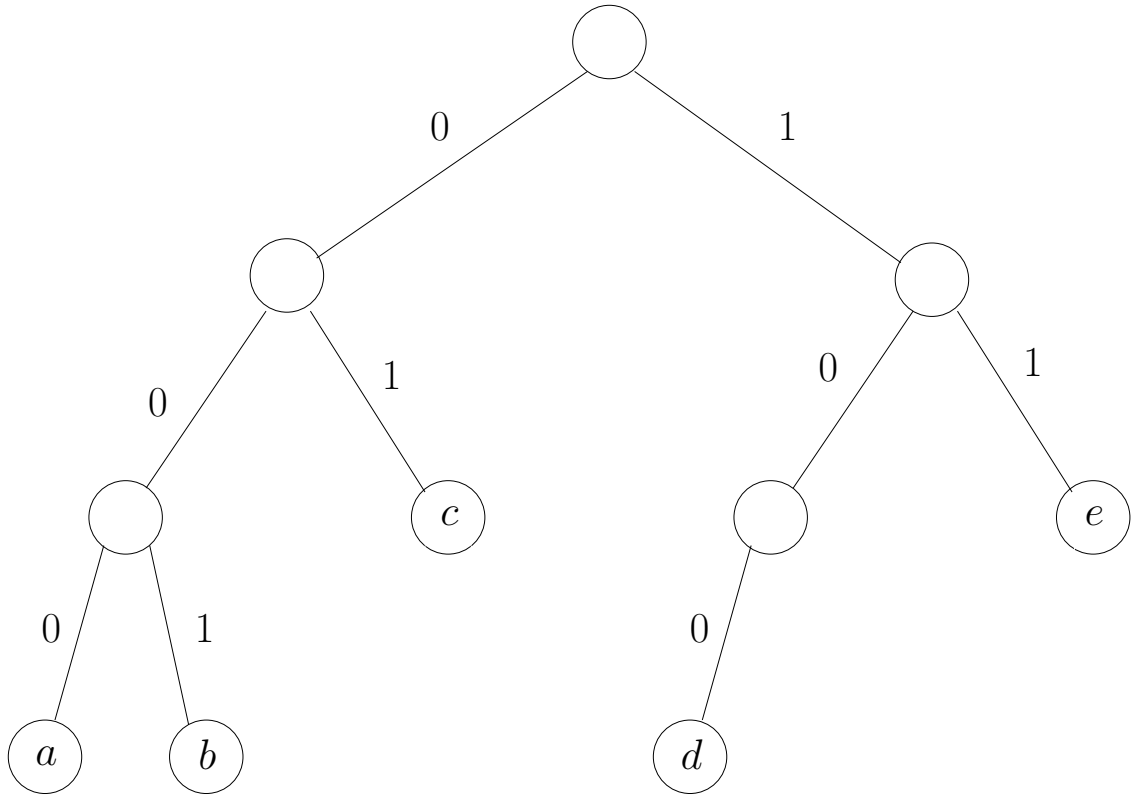


Decode 01001110011111010111010

Decoded as cdejijc

Variable Length Codes as Binary Tree

Symbol	Code
a	000
b	001
c	01
d	100
e	11

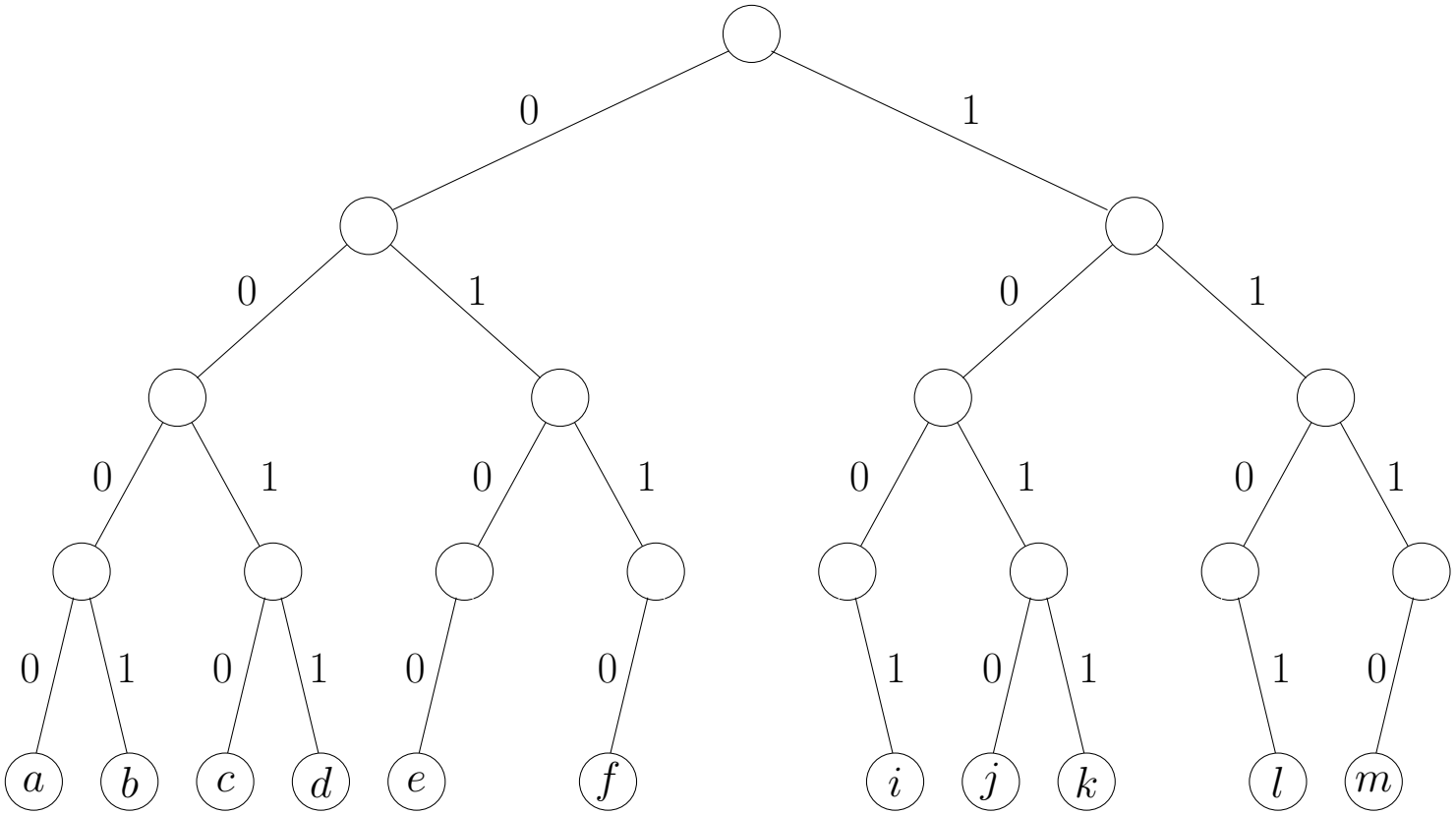


Decode 00011011000111100001000001001

Decoded as aecdcedbacb

Fixed Length Codes as Binary Tree

Symbol	a	b	c	d	e	f	i	j	k	l	m
Code	0000	0001	0010	0011	0100	0110	1001	1010	1011	1010	1110

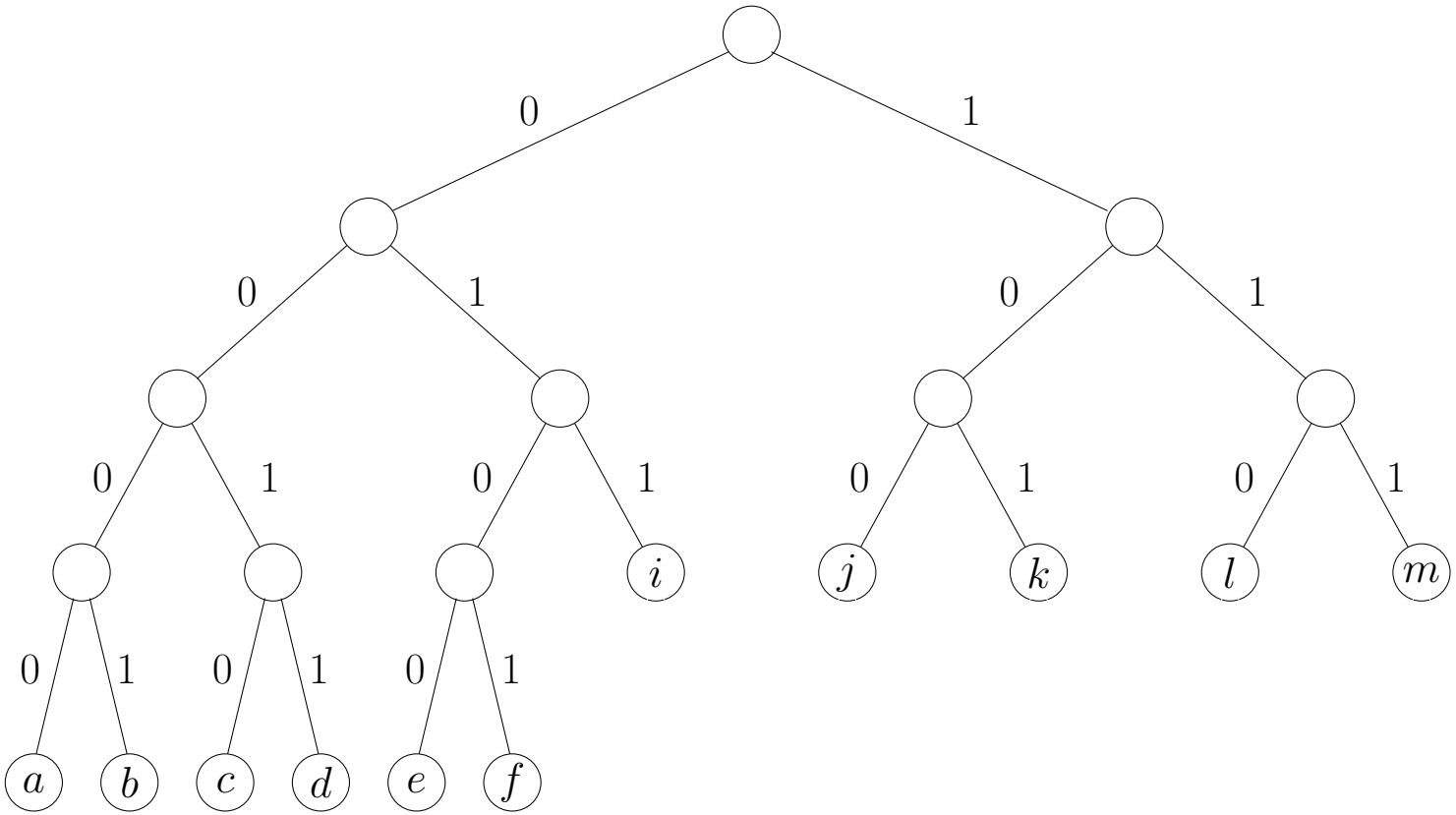


Decode 000001000001001100101010101101100111110

Decoded as aebdclkfim

Variable Length Codes as Binary Tree

Symbol	a	b	c	d	e	f	i	j	k	l	m
Code	0000	0001	0010	0011	0100	0101	011	100	101	110	111

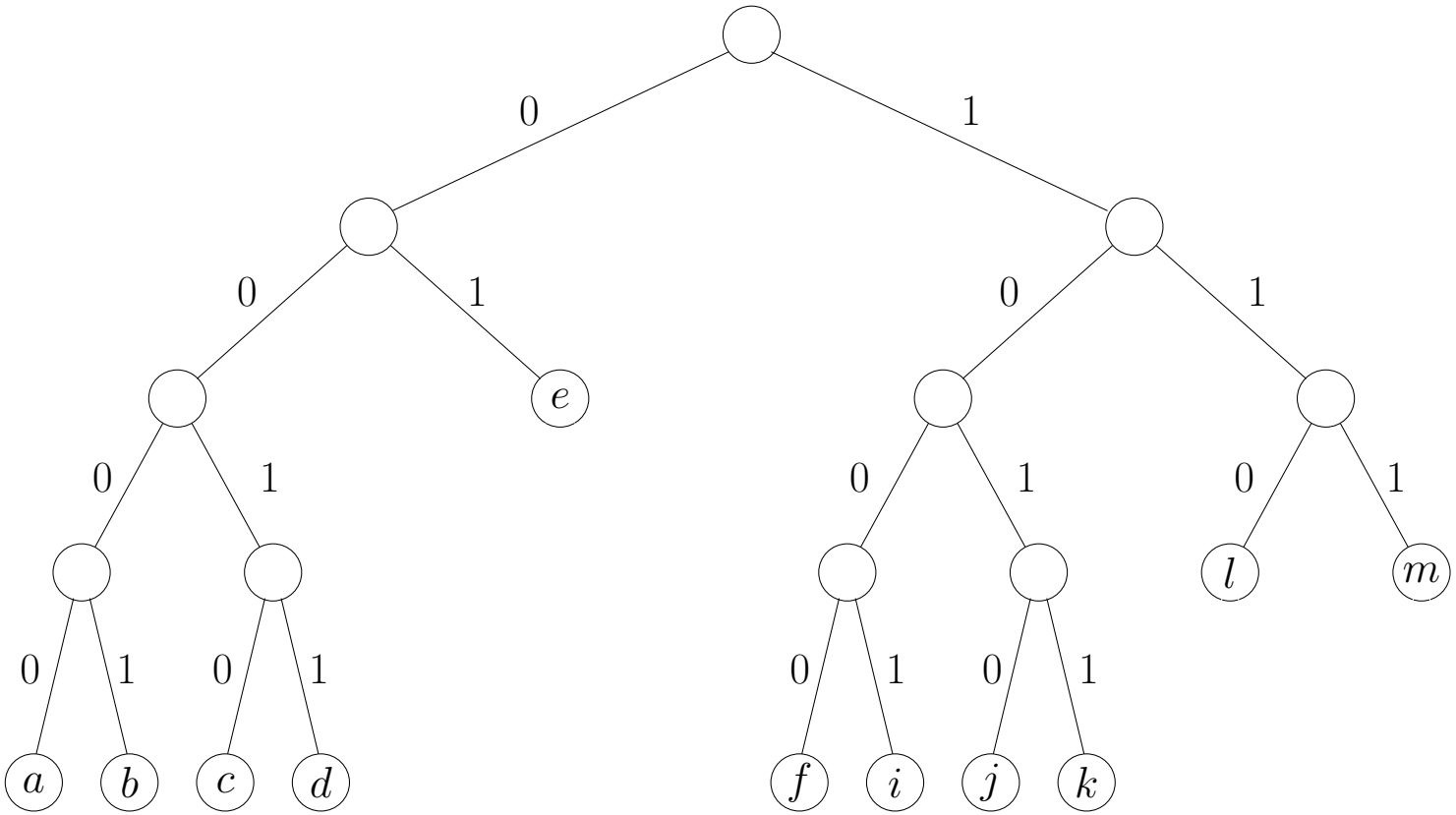


Decode 000001000001001100101101010111111

Decoded as a e b d c l k f i m

Variable Length Codes as Binary Tree

Symbol	a	b	c	d	e	f	i	j	k	l	m
Code	0000	0001	0010	0011	01	1000	1001	1010	1011	110	111



Decode 000000100101100111111001

Decoded as a c e e i m l e

Compression from Codes (Binary Trees)

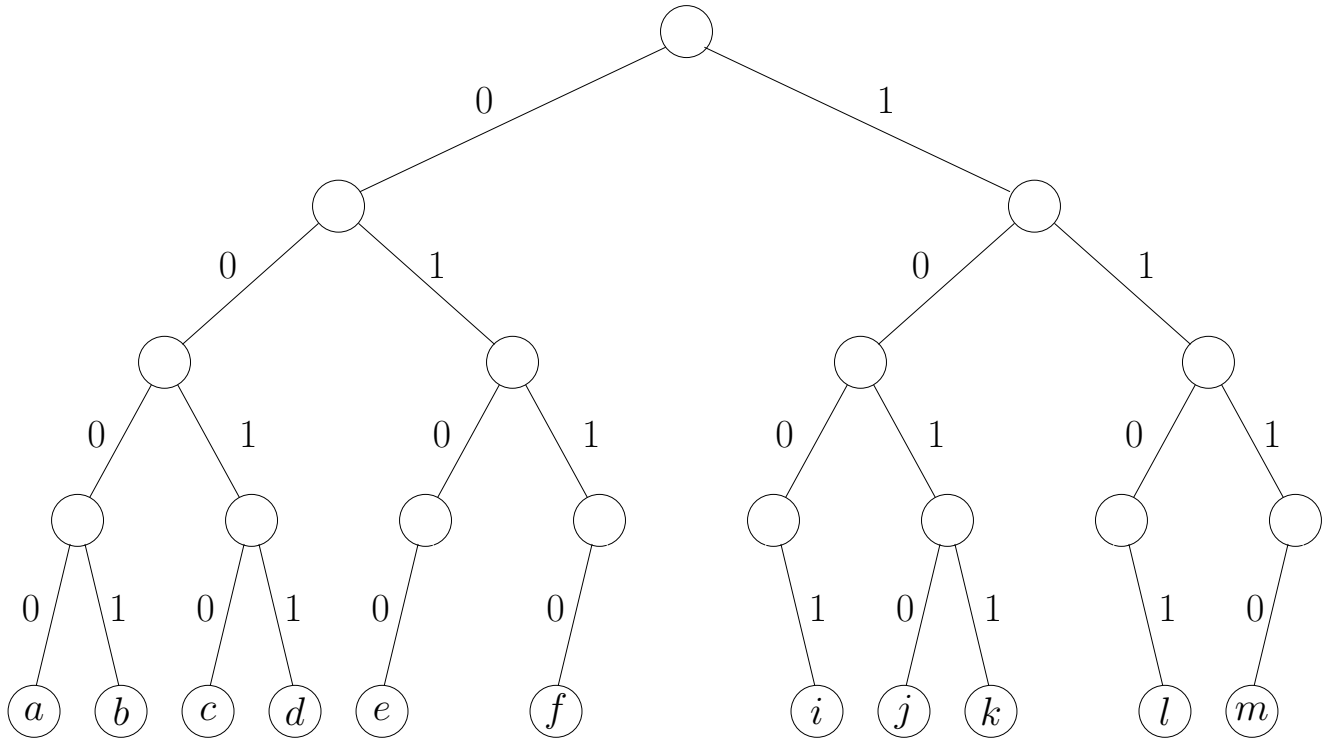
- Alphabet $\Sigma = \{a_1, \ldots, a_n\}$
- A document $D \in \mathcal{D}$
- f: frequency distribution, $f(a_i)$: freq. of a_i in D
- C: a compression scheme with code given as a tree
- $B_C(D) = B(D)$: number of bits to encode D with C
- $C(a_i)$: the code for a_i and $len(C(a_i))$: is its length
- $L(a_i) = len(C(a_i))$= length of root to leaf (a_i) path, $depth(a_i)$

Total number of bits needed to encode the document D is

$$B(D) = \sum_{a_i \in \Sigma} f(a_i)L(a_i) = \sum_{i=1}^{n} f(a_i) \cdot \text{ [depth of } a_i \text{ in } T]$$

Fixed Length Codes as Binary Tree

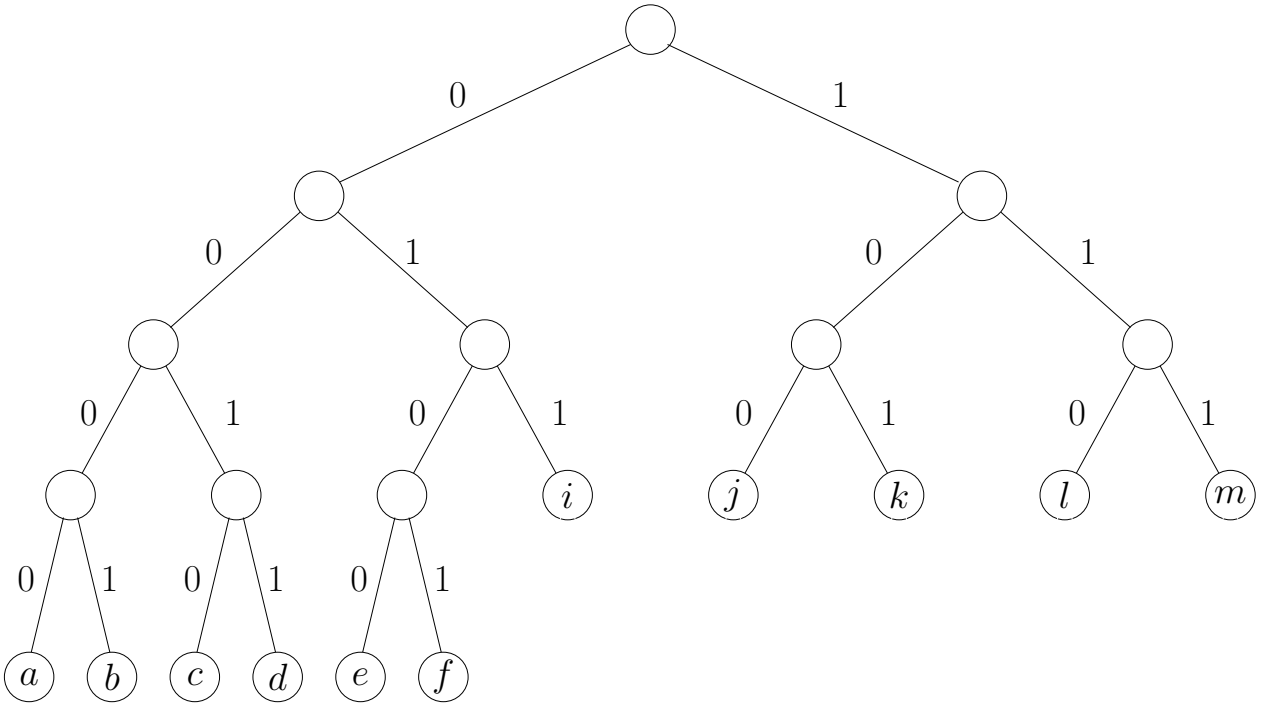
Symbol	a	b	c	d	e	f	i	j	k	l	m
Freq	21	40	6	32	91	7	8	6	23	15	27
Code	0000	0001	0010	0011	0100	0110	1001	1010	1011	1010	1110



$$B(D) = \sum_{a=1}^{n} f(a_i)L(a_i) = 4 \cdot (21 + 40 + 6 + 32 + 91 + 7 + 8 + 6 + 23 + 15 + 27)$$

Variable Length Codes as Binary Tree

Symbol	a	b	c	d	e	f	i	j	k	l	m
Freq	21	40	6	32	91	7	8	6	23	15	27
Code	0000	0001	0010	0011	0100	0101	011	100	101	110	111

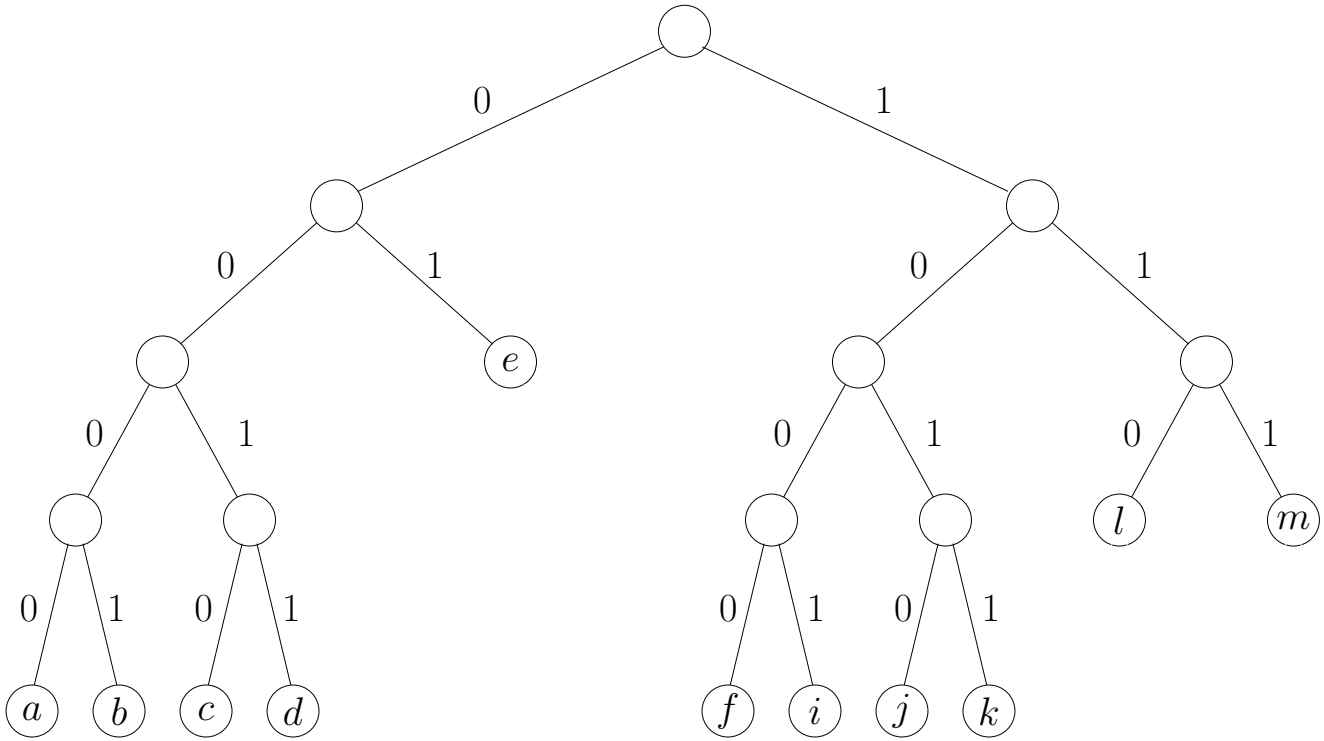


$$B(D) = \sum_{a=1}^{n} f(a_i)L(a_i) =$$

$$21(4) + 40(4) + 6(4) + 32(4) + 91(4) + 7(4) + 8(3) + 6(3) + 23(3) + 15(3) + 27(3)$$

Variable Length Codes as Binary Tree

Symbol	a	b	c	d	e	f	i	j	k	l	m
Freq	21	40	6	32	91	7	8	6	23	15	27
Code	0000	0001	0010	0011	01	1000	1001	1010	1011	110	111



$$B(D) = \sum_{i=1}^{n} f(a_i)L(a_i) =$$

$$21(4) + 40(4) + 6(4) + 32(4) + 91(2) + 7(4) + 8(4) + 6(4) + 23(4) + 15(3) + 27(3)$$