Huffman Codes
Book: Cormen; Section: 16.3

- Data Compression Technique
- Prefix codes (no string is a prefix of another)
- Used for reducing the size of data

Motivation

- Suppose we want to:
  - ➢ Store data in a file
  - ➢ Transmit large files over a network
- Representing each character with a fixed-length code will not result shortest possible file
- Example: 8-bit ASCII code for characters
  – some characters are much more frequent than others
  – using shorter codes for frequent characters and longer ones for infrequent ones will result a shorter file

Example (Page - 429)

	a	**b**	**c**	**d**	**e**	**f**
Frequency (%)	45	13	12	16	9	5
Fixed-length	000	001	010	011	100	101
Variable-length	0	101	100	111	1101	1100

A file of 1,00,000 characters takes:

- $8 \times 1{,}00{,}000 = 8{,}00{,}000$ bits 8-bit ASCII code
- $3 \times 1{,}00{,}000 = 3{,}00{,}000$ bits with fixed-length code
- $(45 \times 1 + 13 \times 3 + 12 \times 3 + 16 \times 3 + 9 \times 4 + 5 \times 4) \times 1000 = 2{,}24{,}000$ bits with variable-length code (25% less)
- With our own code : Additional bits for Table / Tree of code

Example Cntd… (Page - 429)

	a	**b**	**c**	**d**	**e**	**f**
Frequency (%)	45	13	12	16	9	5
Fixed-length	000	001	010	011	100	101
Variable-length	0	101	100	111	1101	1100

Message: abc

→ASCII code : 110000011100001011000011

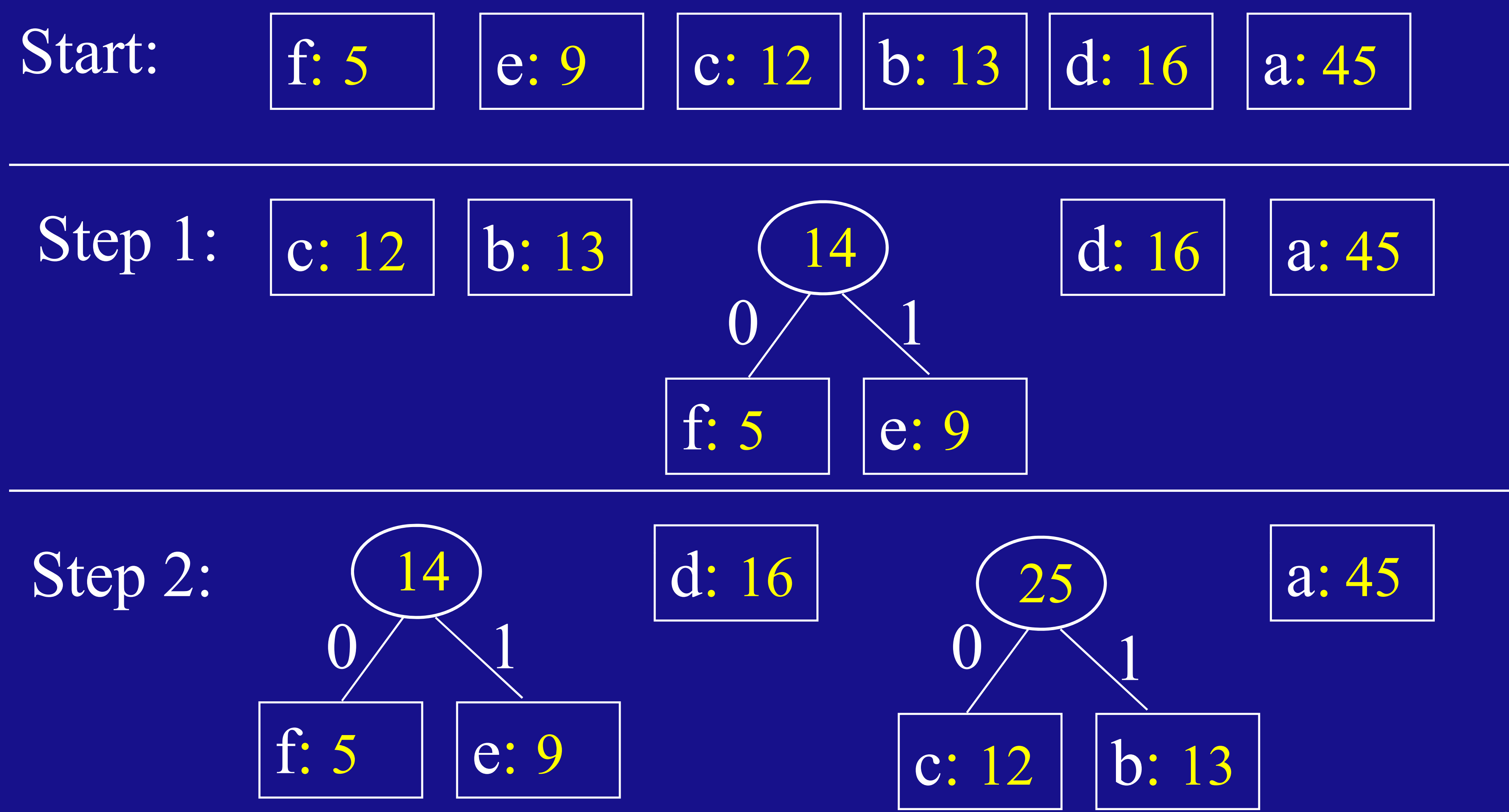
→fixed-length code : 000001010

→variable-length code : 0101100

Huffman code: Construction

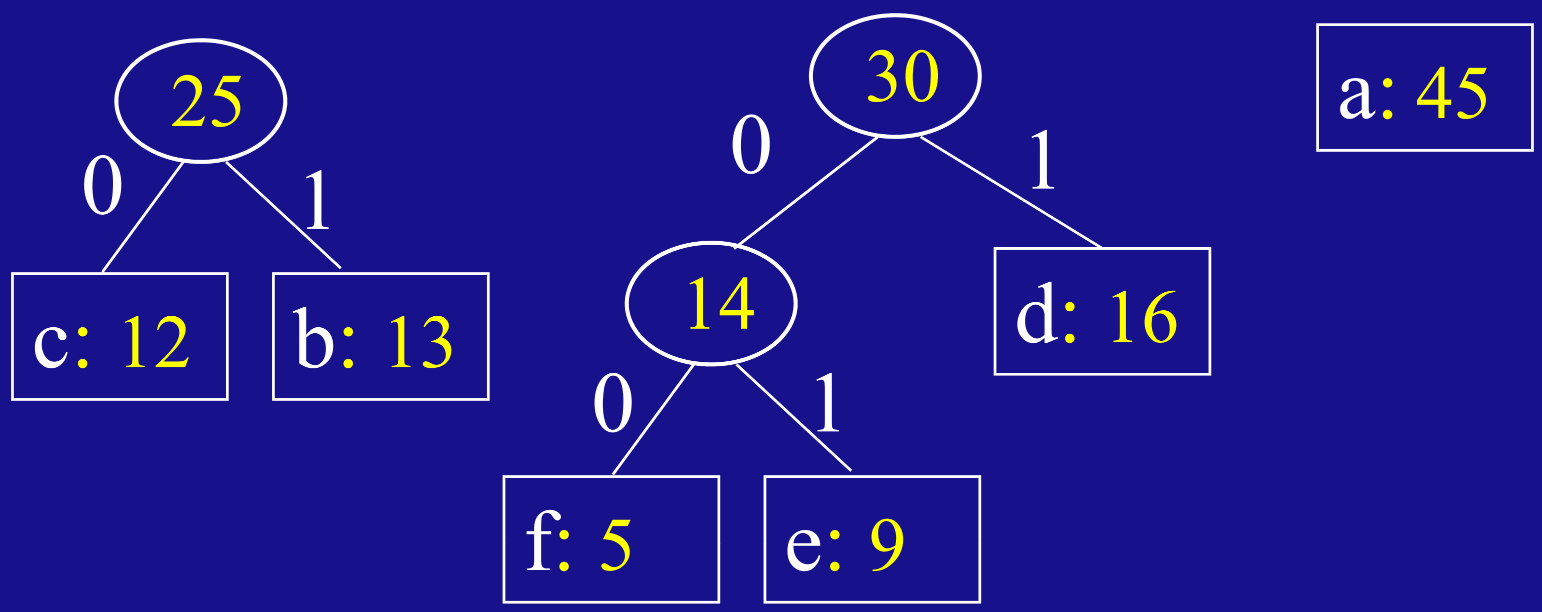
- Build the tree bottom-up, create intermediate nodes by merging the two least-frequent objects.
- To efficiently find the two least-frequent objects, use a minimum priority queue.
- The result of the merger of two objects is a new object whose frequency is the sum of the frequencies of the merged objects.

Example (Page-432) : Huffman code construction(1)



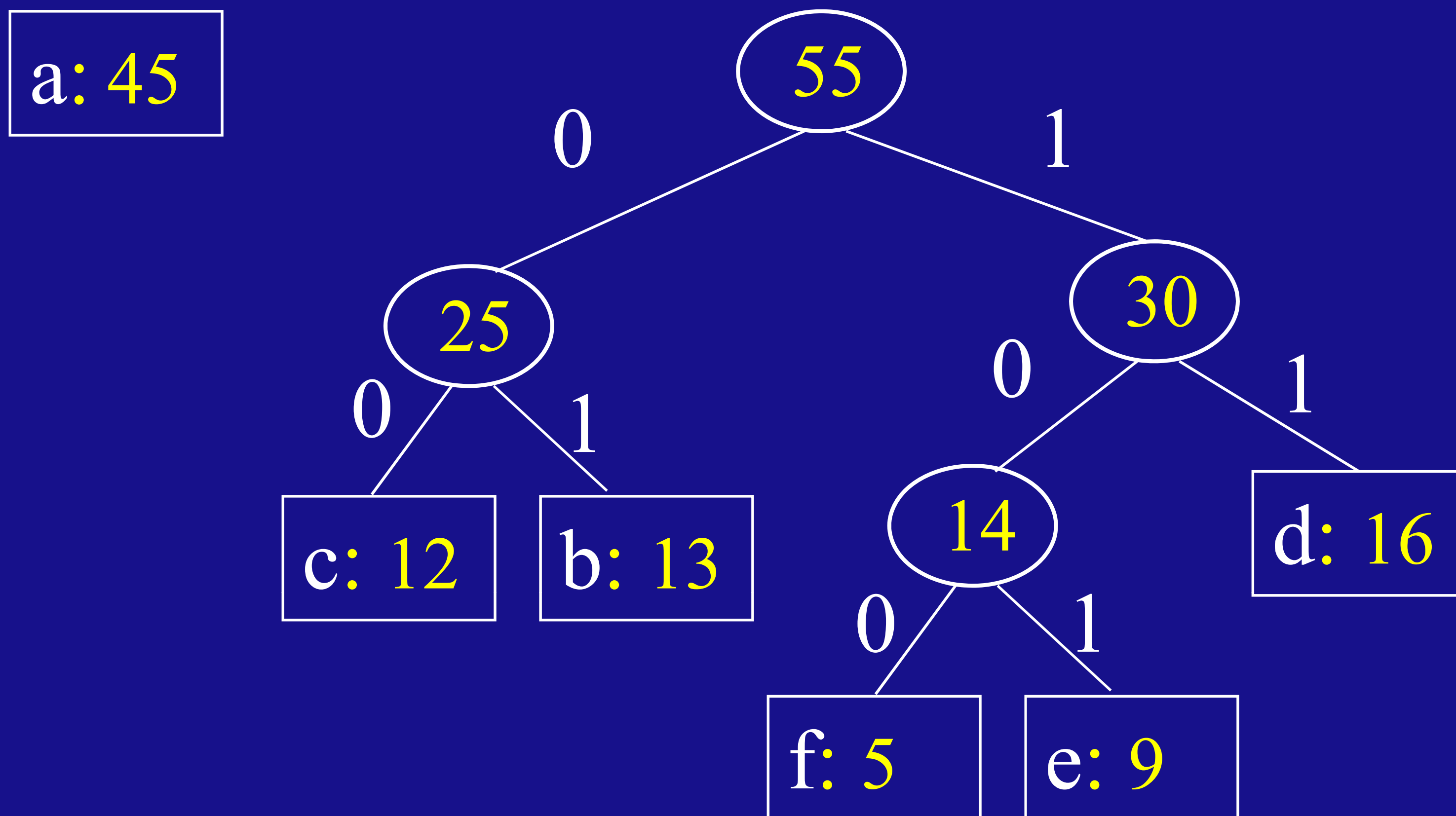
Example (Page-432) : Huffman code construction(2)

Step 3:



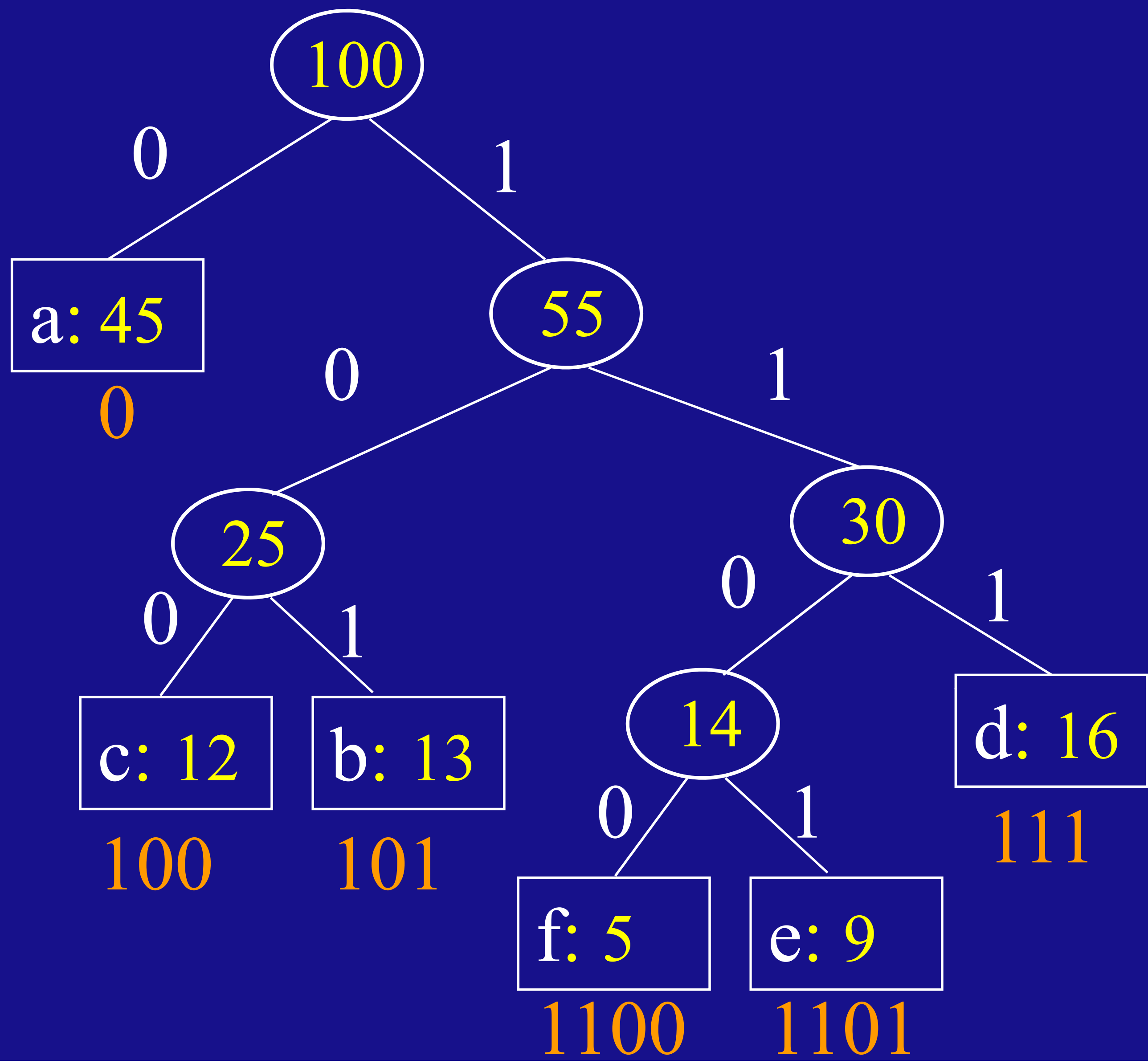
Example (Page-432) : Huffman code construction(3)

Step 4:



Example (Page-432) : Huffman code construction(4)

Step 5:



Huffman code construction algorithm(Page-431)

Huffman(C)

$n \leftarrow |C|$

$Q \leftarrow C$

for $i \leftarrow 1$ **to** $n - 1$

do allocate a new node z

$left[z] \leftarrow x \leftarrow$ Extract-Min(Q)

$right[z] \leftarrow y \leftarrow$ Extract-Min(Q)

$f(z) \leftarrow f(x) + f(y)$

Insert(Q, z)

return Extract-Min(Q)

Huffman code: Decoding

1. Start at the root of the coding tree
2. Read input bits
3. If "0" go left
4. If "1" go right
5. If a leaf node has been reached, output the character stored in the leaf, and return to the root of the tree.

Thank You

Thank You

Stay Safe