SYDE 575: Introduction to Image Processing

Image Compression
Part 2: Variable-rate image compression
(Huffman Coding)

Dealing with quantized DCT coefficients

- After run-length encoding, what do we do with the run-length encoded coefficients?
- Answer: Perform variable-length coding to further reduce the amount of data redundancy in the image
- As such, the data size of the image will vary based on the underlying image characteristics

Recall: Variable Length Coding

- Decrease code length as probability of occurrence increases
- Can achieve much lower coding redundancy by reducing the number of bits needed to store data in an image
- Problem: how do we actually determine what code to use?
  - One set of code may not be well-suited for all images

Data-adaptive Variable Length Coding

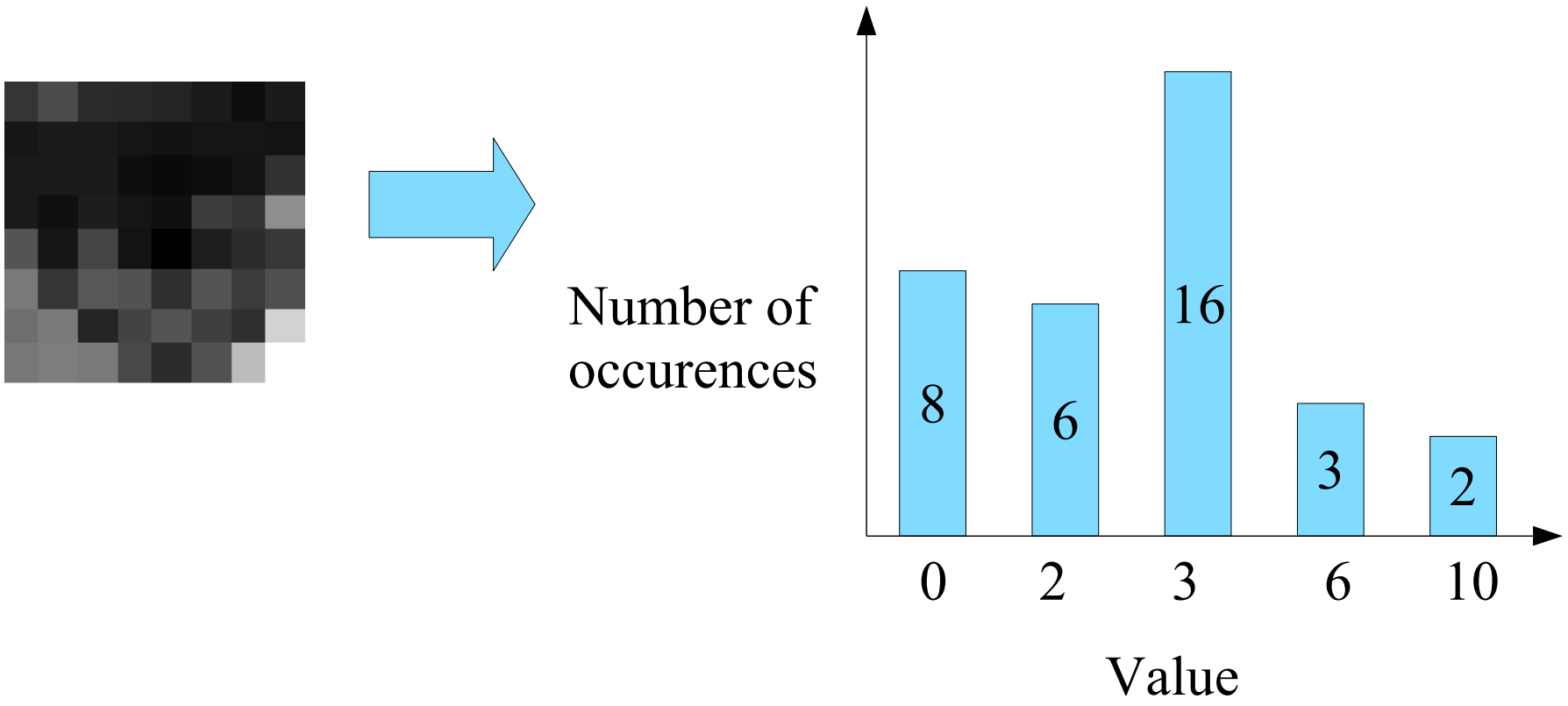
- Idea: change the set of codes used to compress an image based on the underlying image characteristics to achieve better compression specifically for the image
- One of the most popular and commonly used approach: Huffman Coding

Huffman Coding

- Goal: Build minimal length encodings based on frequency of occurrences in the image
- Steps:
  - 1. Determine frequency of occurrences for each possible value in the image
  - 2. Construct Huffman tree
  - 3. Encode image based on codes generated from Huffman tree

Huffman Coding

- 1. Determine frequency of occurrences for each possible value in the image

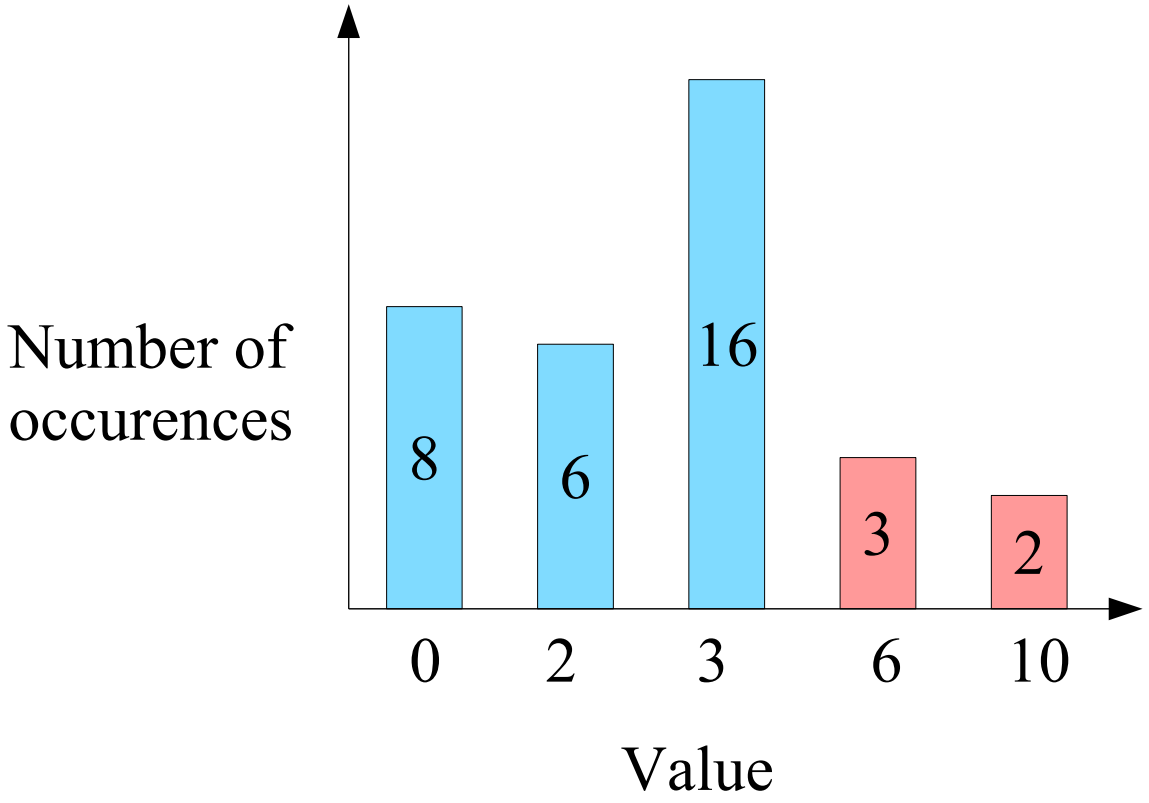


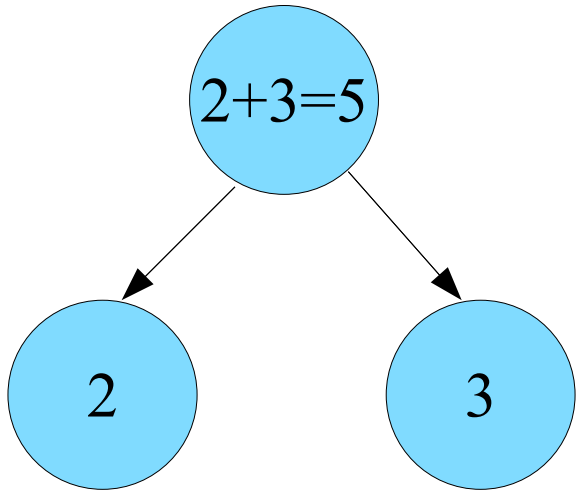
Huffman Coding

- 2. Construct Huffman Tree
  - Huffman Trees are binary trees where:
    - Root node has highest probability of occurrence
    - Lowest leaf nodes have the lowest probability of occurrence
    - Probability of occurrence decreases as we traverse down the tree

Huffman Tree Construction

- Step 1. Take the two lowest frequencies as the leaf nodes and the sum of the frequencies as their parent node



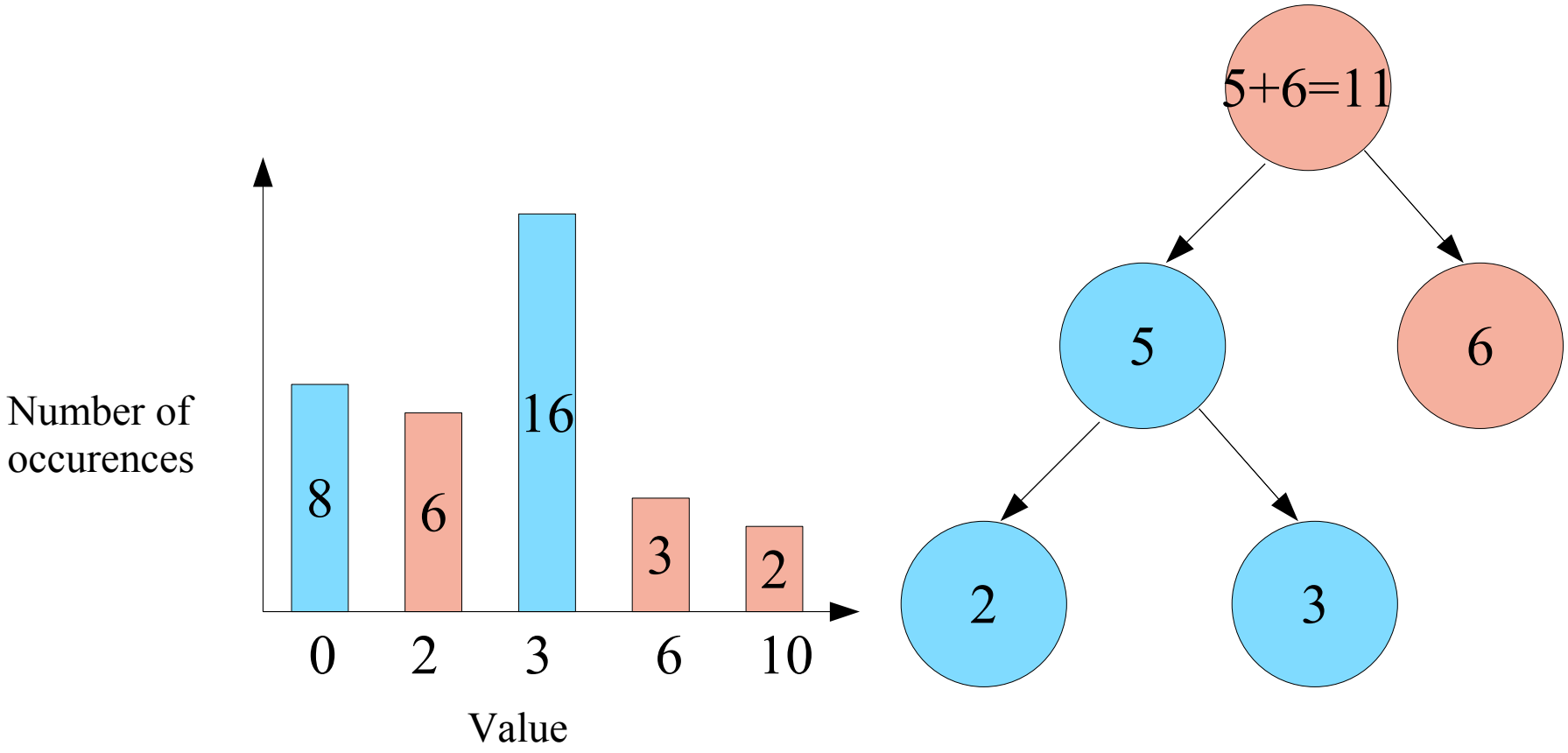


Huffman Tree Construction

- Step 2. Compare value of the parent node with next lowest frequency
  - Lower of the two becomes left child node
  - Higher of the two becomes right child node
  - Sum of the two becomes parent node

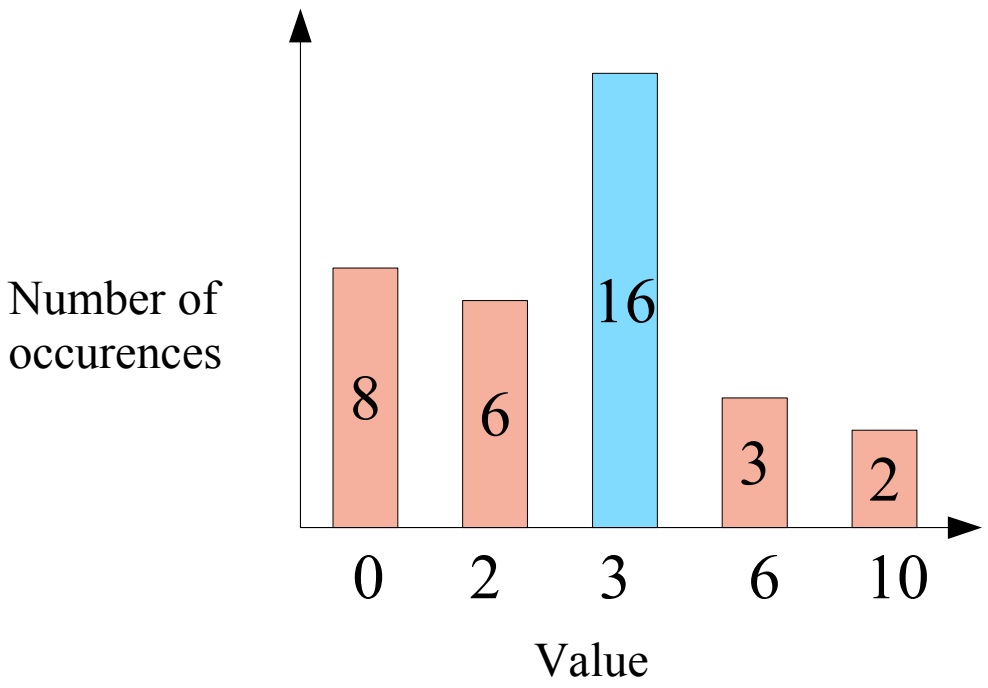
Huffman Tree Construction

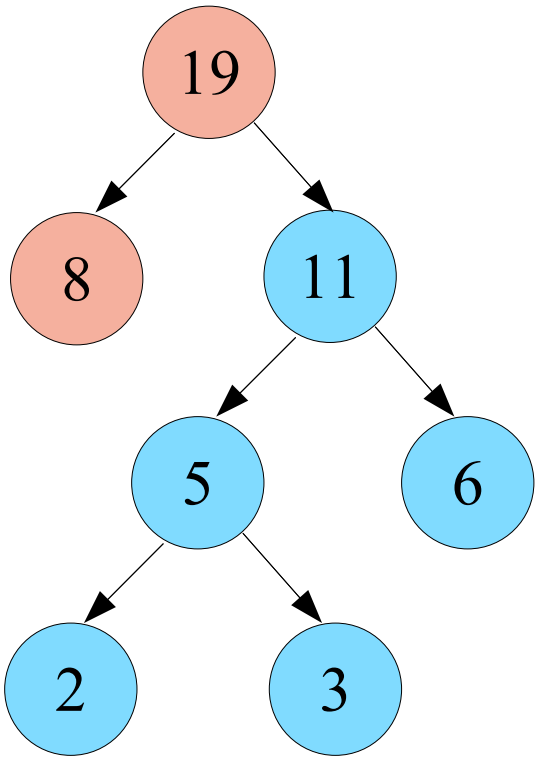
- Step 2.



Huffman Tree Construction

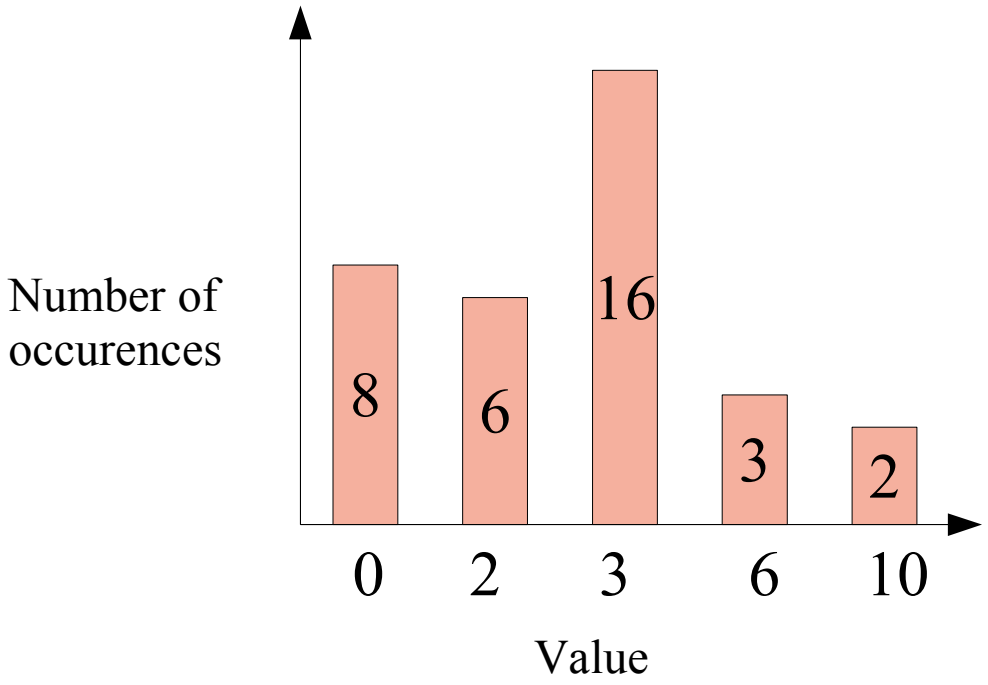
- Step 3. Repeat step 2 until all values have been used

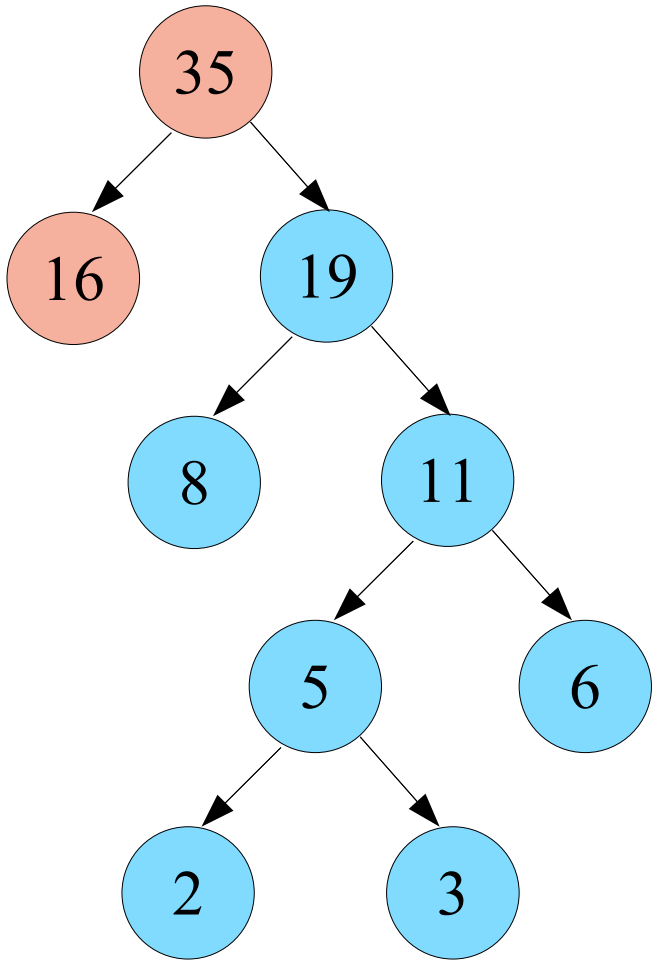




Huffman Tree Construction

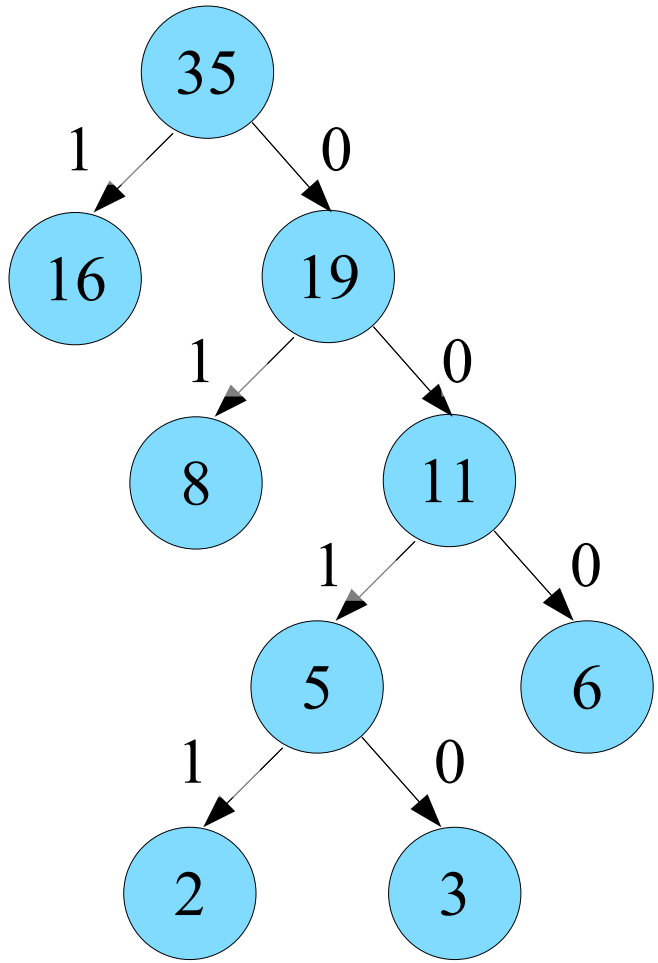
- Step 3. Repeat step 2 until all values have been used





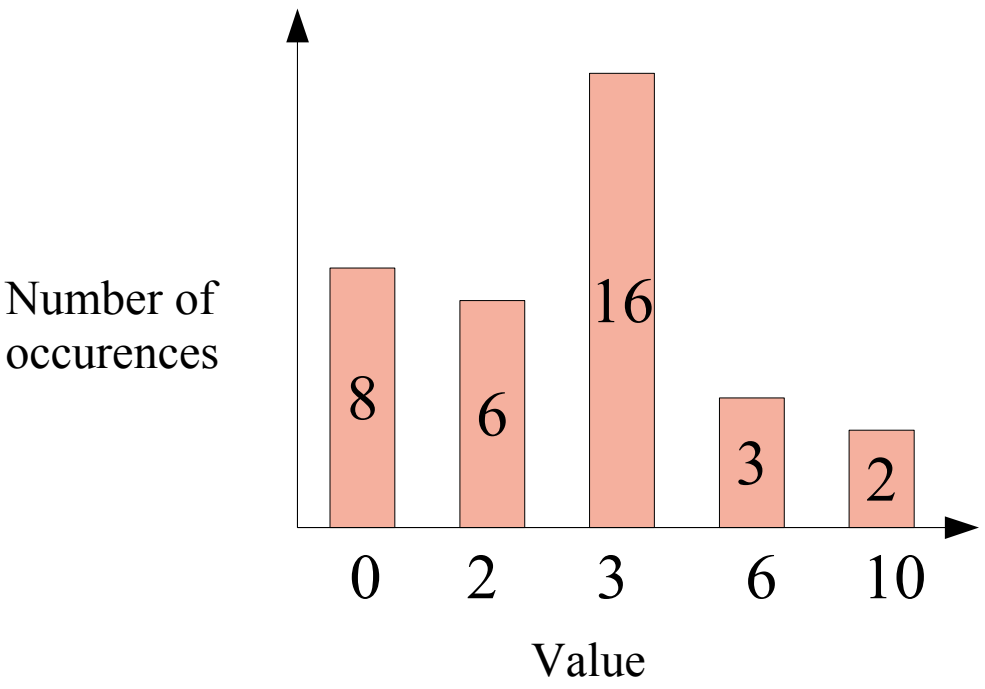
Huffman Tree Construction

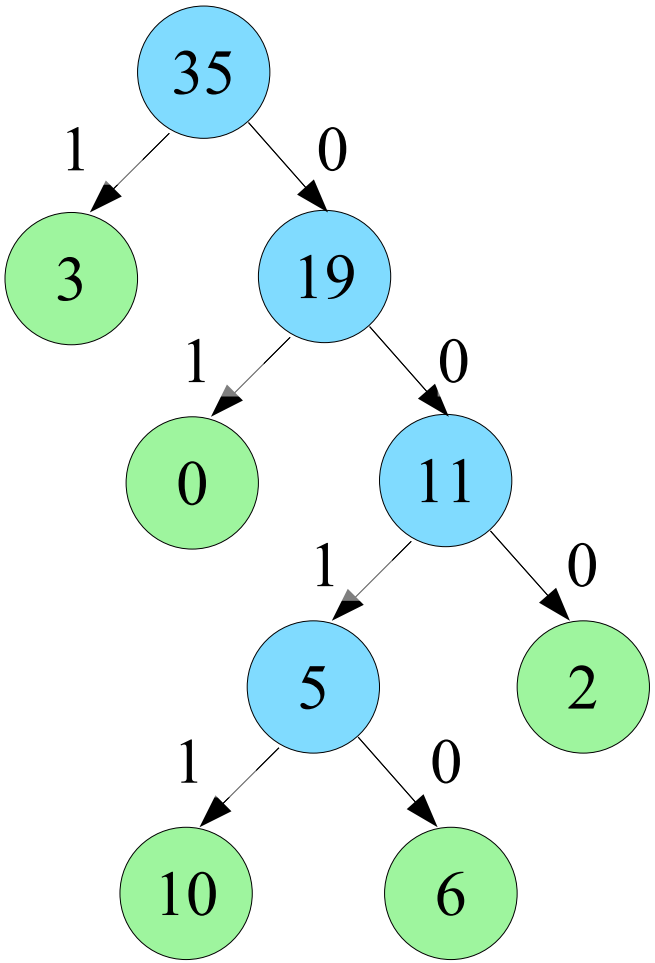
- Step 4. Assign '1' to the left child node and '0' to right child node



Huffman Tree Construction

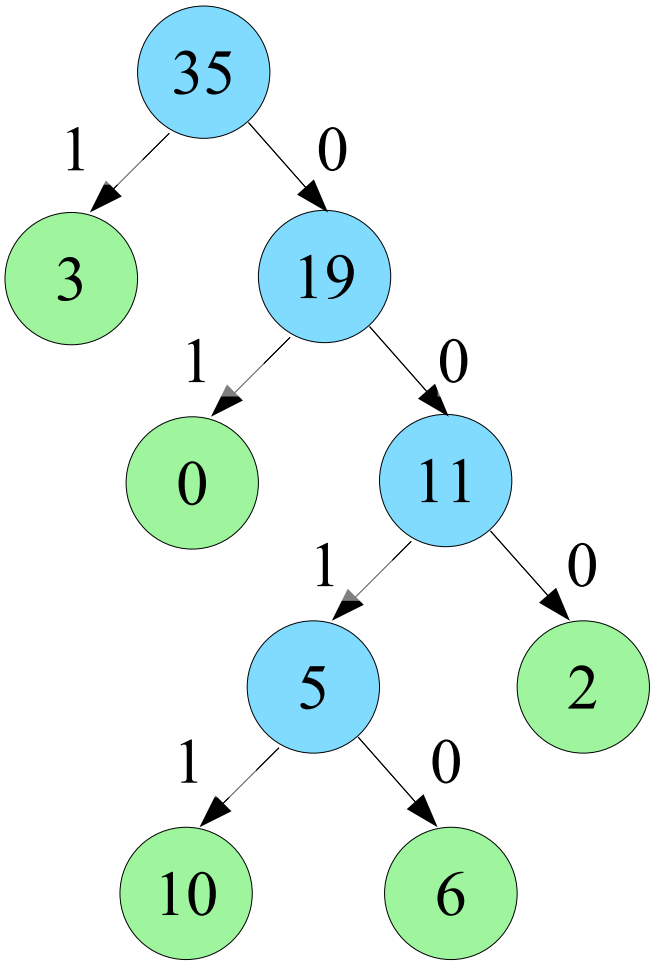
- Step 5. Replace leaf nodes with their corresponding values





Huffman Tree Construction

- Step 6. Compute set of codes from Huffman tree by traversing tree



Value	Code
0	01
2	000
3	1
6	0010
10	0011

Compression Efficiency

- For the above code,

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k)\,pr(r_k)$$

$$= (0.2286)(2)+(0.1714)(3)+(0.4571)(1)+(0.0857)(4)+(0.0571)(4)$$

$$=2$$

- This gives us a compression ratio of:
  - 8 bits per coefficient/2 bits per coefficient = 4:1