Data Structure and Algorithm

Lecture 17

(Huffman Coding)

Huffman Codes

Widely used technique for data compression

Assume the data to be a sequence of characters

Looking for an effective way of storing the data

Binary character code

Uniquely represents a character by a binary string

Fixed-Length Codes

E.g.: Data file containing 100,000 characters

	a	b	c	d	e	f
Frequency (thousands)	45	13	12	16	9	5

3 bits needed

a = 000, b = 001, c = 010, d = 011, e = 100, f = 101

Requires: $100{,}000 \cdot 3 = 300{,}000$ bits

Huffman Codes

Idea:

Use the frequencies of occurrence of characters to build a optimal way of representing each character

	a	b	c	d	e	f
Frequency (thousands)	45	13	12	16	9	5

Variable-Length Codes

E.g.: Data file containing 100,000 characters

	a	b	c	d	e	f
Frequency (thousands)	45	13	12	16	9	5

Assign short codewords to frequent characters and long codewords to infrequent characters

a = 0, b = 101, c = 100, d = 111, e = 1101, f = 1100

$(45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4) \cdot 1{,}000$

= 224,000 bits

Prefix Codes

Prefix codes:

Codes for which no codeword is also a prefix of some other codeword

Better name would be “prefix-free codes”

We can achieve optimal data compression using prefix codes

We will restrict our attention to prefix codes

Encoding with Binary Character Codes

Encoding

Concatenate the codewords representing each character in the file

E.g.:

a = 0, b = 101, c = 100, d = 111, e = 1101, f = 1100

abc = 0 · 101 · 100 = 0101100

Decoding with Binary Character Codes

Prefix codes simplify decoding

No codeword is a prefix of another $\Rightarrow$ the codeword that begins an encoded file is unambiguous

Approach

Identify the initial codeword

Translate it back to the original character

Repeat the process on the remainder of the file

E.g.:

a = 0, b = 101, c = 100, d = 111, e = 1101, f = 1100

001011101 =

$$0 \cdot 0 \cdot 101 \cdot 1101 = \text{aabe}$$

Constructing a Huffman Code

A greedy algorithm that constructs an optimal prefix code called a **Huffman code**

Assume that:

- C is a set of n characters
- Each character has a frequency $f(c)$
- The tree T is built in a bottom up manner
- Left means '0', right means '1'
- More frequent characters will be higher in the tree

Idea:

f: 5	e: 9	c: 12	b: 13	d: 16	a: 45

- Start with a set of $|C|$ leaves
- At each step, merge the two least frequent objects: the frequency of the new node = sum of two frequencies
- Use a min-priority queue Q, keyed on f to identify the two least frequent objects

Constructing a Huffman tree

Alg.: HUFFMAN(C)

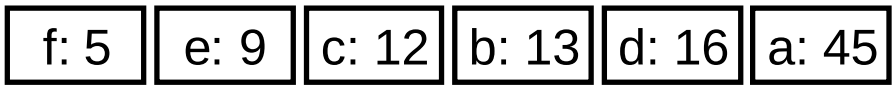
Running time: $O(n \lg n)$

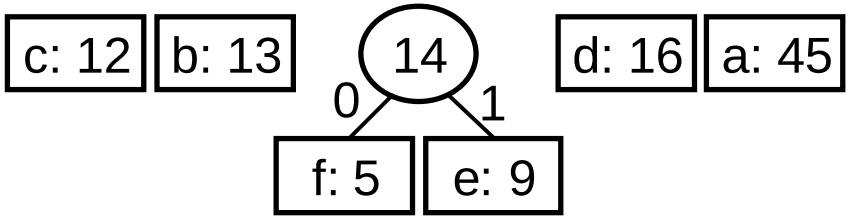
1. $n \leftarrow |C|$
2. $Q \leftarrow C$
3. **for** $i \leftarrow 1$ **to** $n - 1$ ← $O(n)$
4. **do** allocate a new node z
5. left[z] ← x ← EXTRACT-MIN(Q)
6. right[z] ← y ← EXTRACT-MIN(Q)
7. f[z] ← f[x] + f[y]
8. INSERT (Q, z)

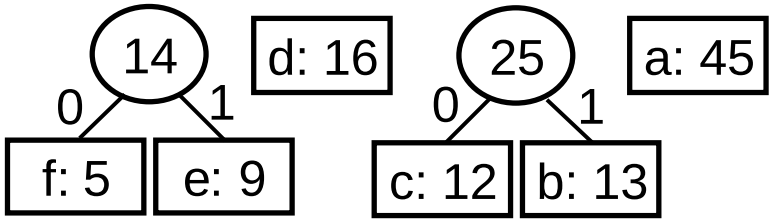
   } $O(\lg n)$
9. **return** EXTRACT-MIN(Q)

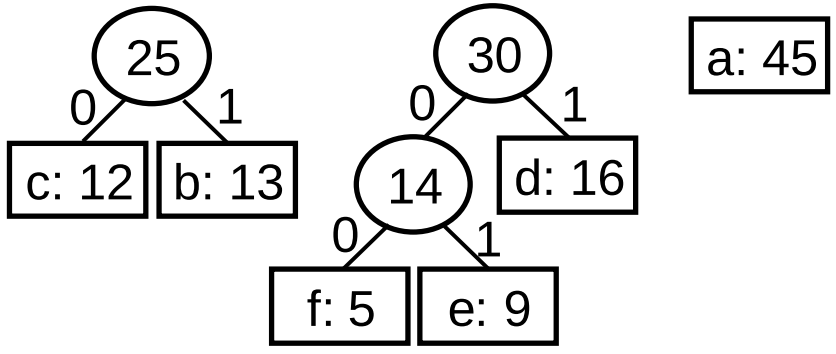
Example

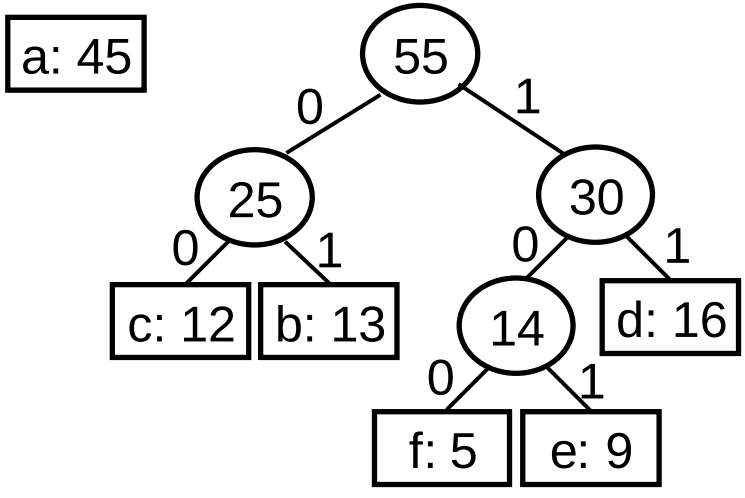
Example

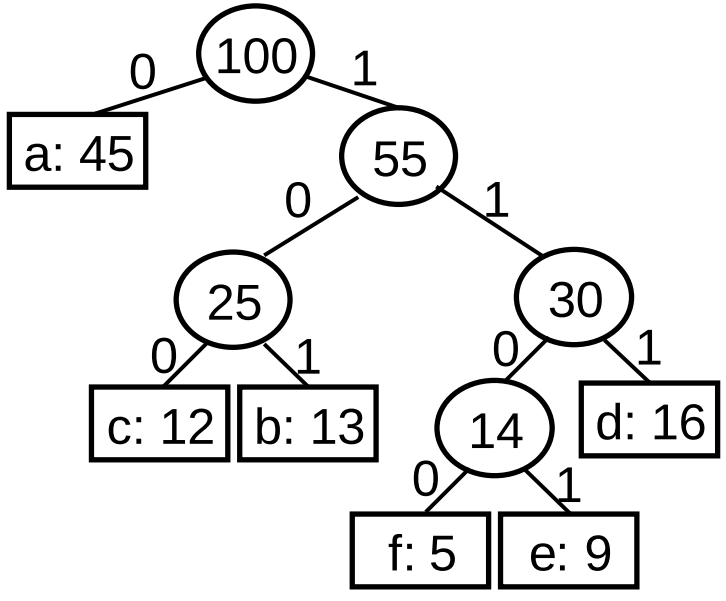

f: 5
e: 9
c: 12
b: 13
d: 16
a: 45


c: 12
b: 13
14
0
1
f: 5
e: 9
d: 16
a: 45


14
0
1
f: 5
e: 9
d: 16
25
0
1
c: 12
b: 13
a: 45


25
0
1
c: 12
b: 13
30
0
1
14
0
1
f: 5
e: 9
d: 16
a: 45


a: 45
55
0
1
25
0
1
c: 12
b: 13
30
0
1
14
0
1
f: 5
e: 9
d: 16


100
0
1
a: 45
55
0
1
25
0
1
c: 12
b: 13
30
0
1
14
0
1
f: 5
e: 9
d: 16

Huffman encoding/decoding using Huffman tree

Encoding:

- Construct Huffman tree using the previous algorithm
- Traverse the Huffman tree to assign a code to each leaf (representing an input character)
- Use these codes to encode the file

Decoding:

- Construct Huffman tree using the previous algorithm
- Traverse the Huffman tree according to the bits you encounter until you reach a leaf node at which point you output the character represented by that leaf node.
- Continue in this fashion until all the bits in the file are read.

Problem types solved by greedy algorithms

- There is no general of knowing whether a problem can be solved by a greedy algorithm.

Conclusion

- Greedy algorithms often lead to polynomial time-solution for an exponential-time problem.