Semigroups and Monoids

Variable-length Codes

Prof.dr. Ferucio Laurențiu Țiplea

Spring 2022

Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Variable-length codes

Huffman codes

Reading and exercise guide

Variable-length codes

Variable-length codes

Definition 1

Let A be a non-empty set. A variable-length code (or simply code) over A is any subset $C \subseteq A^+$ such that C^* is a free sub-monoid of A^*. The elements of C are called code words.

Prove the following equivalent forms of the definition:

1. C is a code over A if any code sequence $w \in C^+$ can be uniquely decomposed into code words
$$w = c_1 \cdots c_n;$$
2. C is a code over A if, for any $u_1, \ldots, u_m, v_1, \ldots, v_n \in C$,
$$u_1 \cdots u_m = v_1 \cdots v_n \quad \Rightarrow \quad n = m \ \wedge \ (\forall i)(u_i = v_i);$$
3. C is a code over A if, for any $u_1, \ldots, u_m, v_1, \ldots, v_n \in C$,
$$u_1 \cdots u_m = v_1 \cdots v_n \quad \Rightarrow \quad u_1 = v_1.$$

Variable-length codes

Example 2

- $C = \{a, ab, ba\}$ is not a code because $aba = (ab)a = a(ba)$;
- $C = \{a, bb, aab, bab\}$ is a code.

Definition 3

1. C is a prefix code if no code word of C is a prefix of any other code word.
2. C is a suffix code if no code word of C is a suffix of any other code word.
3. C is a block code if all code words of C have the same length.

Example 4

- ASCII is a block code.

Variable-length codes

Given a non-empty set $C \subseteq A^+$, define

- $C_1 = \{x \in A^+ | (\exists c \in C)(cx \in C)\}$,
- $C_{i+1} = \{x \in A^+ | (\exists c \in C : cx \in C_i) \vee (\exists c \in C_i : cx \in C)\}$, for any $i \geq 1$.

We get an infinite sequence of sets of words: $C_1, C_2, C_3, \ldots$

Remark 5

If C is finite, then there are i and j such that $i < j$ and $C_i = C_j$.

Theorem 6 (Sardinas-Patterson Theorem)

C is a code over A iff $C \cap C_i = \emptyset$, for any $i \geq 1$.

Proof.

See textbook [1], pages 241-247. □

Variable-length codes

Sardinas-Patterson Algorithm

```
input: finite non-empty set C ⊆ A⁺;
output: code(C) = 1, if C is a code, and code(C) = 0, otherwise;
begin
    C_1 := {x ∈ A⁺ | (∃c ∈ C)(cx ∈ C)};
    if C ∩ C_1 ≠ ∅ then code(C) := 0
        else begin
                i := 1; cont := 1;
                while cont = 1 do
                   begin
                      i := i + 1;
                      C_i := {x ∈ A⁺ | (∃c ∈ C_{i−1})(cx ∈ C) ∨ (∃c ∈ C)(cx ∈ C_{i−1})};
                      if C ∩ C_i ≠ ∅ then begin code(C) := 0; cont := 0 end
                          else if (∃j < i)(C_i = C_j)
                                  then begin code(C) := 1; cont := 0 end;
                   end;
             end;
end.
```

Huffman codes

Huffman codes

- Have been proposed by David Huffman in 1952;
- Are used to encode information sources (an information source is a device which outputs symbols from a given alphabet according to certain probabilities depending, in general, on preceding choices as well as the particular symbol in question);
- Are prefix codes of minimum length among all the prefix codes associated to a given information source;
- Associate short code words to highly probable symbols (which appear more frequently), and longer code words to symbols with smaller probabilities.

Information source

Definition 7

An information source is a couple $IS = (A, \pi)$, where A is a non-empty and at most countable set, called the source alphabet, and π is a probability distribution on A.

Only finite information sources will be considered. See the textbook for extension to countable information sources.

Definition 8

Let $IS = (A, \pi)$ be an information source and $h : A \to \Sigma^*$ be a homomorphism. The (average) length of h with respect to IS is

$$L_h(IS) = \sum_{a \in A} |h(a)|\pi(a) .$$

Encoding of an information source

Definition 9

Let $IS = (A, \pi)$ be an information source and $h : A \to \Sigma^*$ be a homomorphism. h is called a code or encoding of IS if $C = \{h(a) | a \in A\}$ is a code.

Example 10

Let IS be the information source

A	a	b	c	d	e	f
π	0.4	0.1	0.1	0.1	0.2	0.1

and h be the encoding

A	a	b	c	d	e	f
h	0	1100	1101	1110	10	1111

Then, the length of h w.r.t. IS is $L_h(IS) = 2.4$. That is, on average, 2.4 bits are needed to encode any symbol of the information source.

Huffman codes

Definition 11

Let $IS = (A, \pi)$ be an information source and $h : A \to \Sigma^*$ be an encoding for IS. h is called a Huffman code or a Huffman encoding of IS if:

- $C = \{h(a) | a \in A\}$ is a prefix code;
- h has minimum length among all the prefix codes of IS.

Given an information source IS, are there Huffman encodings for IS ?

The answer is positive.

Huffman algorithm

1. Let IS be an information source with $n \geq 2$ symbols

A	a_1	a_2	$\cdots$	a_{n-1}	a_n
π	p_1	p_2	$\cdots$	p_{n-1}	p_n

   where $p_1 \geq p_2 \geq \cdots \geq p_{n-1} \geq p_n$;

2. If $n = 2$, then $h(a_1) = 0$ and $h(a_2) = 1$ (or vice-versa) is a Huffman code for IS;

3. If $n \geq 3$, then compute a reduced source IS' for IS

A'	a_1	a_2	$\cdots$	a_{n-2}	$a_{n-1,n}$
π'	p_1	p_2	$\cdots$	p_{n-2}	$p_{n-1,n}$

   where $p_{n-1,n} = p_{n-1} + p_n$;

Huffman algorithm

4. If h' is a Huffman code for IS', then h given by

$$h(x) = \begin{cases} h'(x), & \text{if } x \notin \{a_{n-1}, a_n\} \\ h'(x)0, & \text{if } x = a_{n-1} \\ h'(x)1, & \text{if } x = a_n, \end{cases}$$

   is a Huffman code for IS.

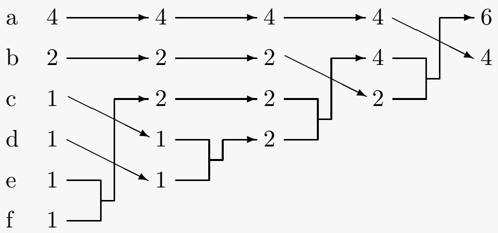
Example of Huffman encoding

Example 12

Let *IS* be the following information source:

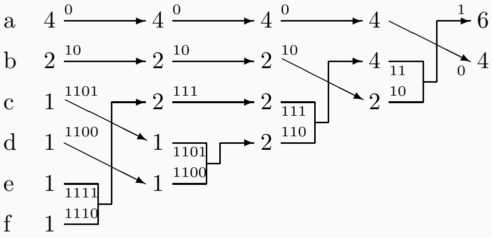
A	a	b	c	d	e	f
π	0.4	0.2	0.1	0.1	0.1	0.1

Compute a sequence of reduced sources for *IS*:



Example of Huffman encoding

Assign codes to each reduced source from right to left:



The Huffman code is $h(a) = 0$, $h(b) = 10$, $h(c) = 1101$, $h(d) = 1100$, $h(e) = 1111$, and $h(f) = 1110$. The length of h is 2.4. It is the minimum length code among all the prefix codes associated to IS.

Data compression

Huffman codes can be used to compress data as follows. Let α be a text:

1. parse α and, for each symbol a in α compute its number of occurrences;
2. let IS be the information source thus obtained. Compute a Huffman code h for IS;
3. encode α by $h(\alpha)$ (obtained by replacing each symbol a in α by $h(a)$).

Compression ratio = is the ratio of the size of the original data to the size of the compressed data.

Compression rate = is the rate of the compressed data (typically, it is in units of bits/sample, bits/character, bits/pixels, or bits/second).

Data compression

There are two types of data compression:

- lossless data compression – allows the exact original data to be reconstructed from the compressed data;
- lossy data compression – does not allow the exact original data to be reconstructed from the compressed data.

Data compression by Huffman codes is lossless!

Is there any limit to lossless data compression?

The answer is positive. The limit is called the entropy. The exact value of the entropy depends on the (statistical nature of the) information source. It is possible to compress the source, in a lossless manner, with compression rate close to its entropy. It is mathematically impossible to do better than that.

Huffman codes

Definition 13

Let S be an IS with n symbols and probabilities $p_1, \ldots, p_n$. The entropy of S, denoted $H(S)$ or $H(p_1, \ldots, p_n)$, is defined by

$$\boxed{H(S) = \sum_{i=1}^{n} p_i \log(1/p_i)}$$

(mathematical convention: $0 \cdot \log(1/0) = 0$).

The entropy was introduced in computer science by Claude Shannon [6], who is now considered the father of information theory.

Huffman codes

Proposition 14

For any probability distribution $p_1, \ldots, p_n$, the following hold:

1. $0 \leq H(p_1, \ldots, p_n) \leq \log n$;
2. $H(p_1, \ldots, p_n) = 0$ *iff* $p_i = 1$, *for some* i;
3. $H(p_1, \ldots, p_n) = \log n$ *iff* $p_i = 1/n$, *for any* i.

Proof.

See textbook [1], pages 261-266. $\square$

Definition 15

Let $S_1 = (\{a_i | 1 \leq i \leq n\}, (p_i | 1 \leq i \leq n))$ and $S_2 = (\{b_j | 1 \leq j \leq m\}, (q_j | 1 \leq j \leq m))$ be two ISs. The product of S_1 and S_2, denoted $S_1 \circ S_2$, is the IS

$$S_1 \circ S_2 = (\{(a_i, a_j) | 1 \leq i \leq n,\ 1 \leq j \leq m\}, (p_i \cdot q_j | 1 \leq i \leq n,\ 1 \leq j \leq m)).$$

Huffman codes

Prove the following properties:

Proposition 16

For any finite information sources S_1 *and* S_2*,*

$$\boxed{H(S_1 \circ S_2) = H(S_1) + H(S_2)}\,.$$

Composing S with itself k times, we obtain the source S^k. Then,

$$\boxed{H(S^k) = kH(S)}\,.$$

Huffman codes

Theorem 17 (Shannon's noiseless coding theorem)

Let S be an information source. Then:

(1) $H(S) \leq L_h(S)$, *for any code h of S;*

(2) $H(S) \leq L_h(S) < H(S) + 1$, *for any Huffman code h of S;*

(3) $lim_{k\to\infty} \dfrac{L_{min}(S^k)}{k} = H(S)$, *where $L_{min}(S')$ is the average length of some Huffman code for S'.*

Proof.

See textbook [1], pages 266-267. □

Shannon's noiseless coding theorem places a lower bound on the minimal possible expected length of an encoding of a source S, as a function of the entropy of S.

Adaptive Huffman coding

Huffman encoding for an input $w \in \Sigma^+$ requires two steps:

- determine the frequency of occurrences of each letter a in w;
- design a Huffman code for Σ w.r.t. the probability distribution

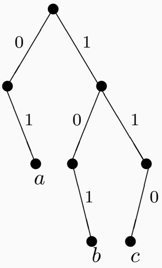
$$p_a = \frac{\text{frequency of } a \text{ in } w}{|w|}.$$

Then, encode w by this Huffman code.

Because this procedure requires two parsings of the input, it is time-consuming for large inputs (although the compression rate by such an encoding is optimal). In practice, an alternative method which requires only one parsing of the input is used. It is called the adaptive Huffman coding [3, 4, 5, 7].

Tree representation of binary codes

A useful graphical representation of a finite code $C \subseteq A^+$ consists of a tree with nodes labeled by symbols in A such that the code words are exactly the sequences of labels collected from the root to leaves. For example, the tree below is the graphical representation of the prefix code $\{01, 110, 101\}$, where a is encoded by 01, b by 101, and c by 110.



The sibling transformation

The encoding of an input w by the adaptive Huffman technique is based on the construction of a sequence of Huffman trees as follows:

- start initially with a Huffman tree $\mathcal{T}_0$ associated to the alphabet A (each symbol of A has frequency 1);
- if $\mathcal{T}_n$ is the current Huffman tree and the current input symbol is a (that is, $w = uav$ and u has been already processed), then output the code of a in $\mathcal{T}_n$ (this code is denoted by $code(a, \mathcal{T}_n)$) and update the tree $\mathcal{T}_n$ by applying to it the sibling transformation; the new tree is denoted $\mathcal{T}_{n+1}$.

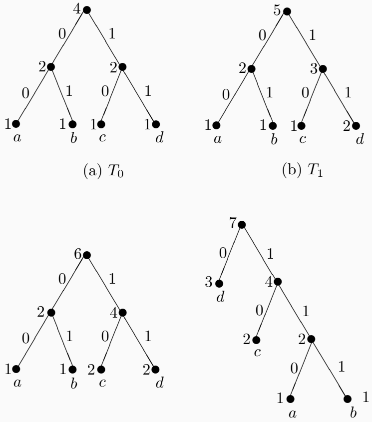
The sibling transformation

The sibling transformation applied to symbol a and tree $\mathcal{T}_n$ consists of:

1. compare a to its successors in the tree (from left to right and from bottom to top). If the immediate successor has frequency $k+1$ or greater, where k is the frequency of a in $\mathcal{T}_n$, then the nodes are still in sorted order and there is no need to change anything. Otherwise, a should be swapped with the last successor which has frequency k or smaller (except that a should not be swapped with its parent);
2. increment the frequency of a (from k to $k+1$);
3. if a is the root, the loop halts; otherwise, the loop repeats with the parent of a.

Adaptive Huffman coding

A sequence of Huffman trees used to encode the string *dcd* over the alphabet $\{a, b, c, d\}$:



(a) T_0 (b) T_1

(d) T_2 (c) T_3

Time-varying codes

Huffman adaptive is not a variable-length code! The same character may be encoded by different code words!

Huffman adaptive is a time-varying code! For more details regarding time-varying codes see [2].

Reading and exercise guide

Reading and exercise guide

It is highly recommended that you do all the exercises marked in red from the slides.

Course readings:

1. Pages 235-267 from textbook [1].

References

[1] Ferucio Laurențiu Țiplea. *Algebraic Foundations of Computer Science*. "Alexandru Ioan Cuza" University Publishing House, Iași, Romania, second edition, 2021.

[2] Ferucio Laurențiu Țiplea, Erkki Mäkinen, Dragos Trincă, and Costel Enea. Characterization results for time-varying codes. *Fundamenta Informaticae*, 53(2):185–198, may 2002.

[3] Newton Faller. An adaptive system for data compression. In *7th Asilomar Conference on Circuits, Systems and Computers*, pages 593–597. IEEE, 1973.

[4] Robert Gallager. Variations on a theme by huffman. *IEEE Transactions on Information Theory*, 24(6):668–674, 1978.

[5] Donald E Knuth. Dynamic Huffman coding. *Journal of Algorithms*, 6(2):163–180, 1985.

[6] Claude Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[7] Jeffrey Scott Vitter. Design and analysis of dynamic huffman codes. *J. ACM*, 34(4):825–845, Oct 1987.