UNIVERSITY OF LIVERPOOL

Department of Computer Science
Sebastian Wild

Date: 2020-03-25
Version: 2020-04-07 13:50

Tutorial 7 for COMP 526 – Applied Algorithmics, Winter 2020

—including solutions—

It is highly recommended that you first try to solve the problems on your own before consulting the sample solutions provided below.

Problem 1 (Move-to-front transform)

Let $S = (20, 30, 30, 20, 40, 30, 20, 20, 20)$ be an input sequence of numbers whose values are initially stored in the list $Q = [20, 30, 40]$. Build an output sequence and trace the content of Q throughout the execution of *MTF (Move-to-Front) algorithm*.

Solutions for Problem 1 (Move-to-front transform)

The MTF encoding traverses the input sequence character by character. It replaces each value by its current position in the list and then moves this value to the front of the list. In case of input sequence S and initial content of Q the transformation process is as follows:

0)	**[20,30,40]**	0	5)	[40, 20, 30]	2
1)	[20, 30, 40]	1	6)	[30, 40, 20]	2
2)	[30, 20, 40]	0	7)	[20, 30, 40]	0
3)	[30, 20, 40]	1	8)	[20, 30, 40]	0
4)	[20, 30, 40]	2	9)	**[20,30,40]**	

Note how the second occurrence of the repeated substring 30, 20 leads to a run (of twos).

Problem 2 (Lempel-Ziv-Welch compression)

Given word $w = \texttt{ASNXASNASNA}$ over the ASCII character set (relevant parts of ASCII are provided on the right). Construct, step by step, the Lempel-Ziv-Welch (LZW) factorization of w (i.e., the phrases encoded by one codeword) and provide the compressed representation of w; it suffices to show the encoded text C using integer numbers (no need for binary encodings).

Code	Character
65	A
...	...
78	N
...	...
83	S
...	...
88	X
...	...

Solutions for Problem 2 (Lempel-Ziv-Welch compression)

Steps of the algorithm:

Phrase	$C[i]$	New code	$s + c$
A	65	128	AS
S	83	129	SN
N	78	130	NX
X	88	131	XA
AS	128	132	ASN
N	78	133	NA
ASN	132	134	ASNA
A	65	135	

The compressed representation is $C = 65, 83, 78, 88, 128, 78, 132, 65$.

The LZW encoding decomposes w into phrases as follows: $w = \texttt{A}|\texttt{S}|\texttt{N}|\texttt{X}|\texttt{AS}|\texttt{N}|\texttt{ASN}|\texttt{A}$.

Problem 3 (No Free Lunch)

Prove the following *no-free-lunch* theorems for lossless compression.

1. *Weak version:* For every compression algorithm A and $n \in \mathbb{N}$ there is an input $w \in \Sigma^n$ for which $|A(w)| \geq |w|$, i. e. the "compression" result is no shorter than the input.

   Hint: Try a proof by contradiction. There are different ways to prove this.

2. *Strong version:* For every compression algorithm A and $n \in \mathbb{N}$ it holds that

   $$\left|\{w \in \Sigma^{\leq n} : |A(w)| < |w|\}\right| < \tfrac{1}{2} \cdot \left|\Sigma^{\leq n}\right| .$$

   In words, less than half of all inputs of length at most n can be compressed below their original size.

   Hint: Start by determining $|\Sigma^{\leq n}|$.

The theorems hold for every non-unary alphabet, but you can restrict yourself to the binary case, i.e., $\Sigma = \{\texttt{0}, \texttt{1}\}$.

We denote by $\Sigma^\star$ the set of all (finite) strings over alphabet Σ and by $\Sigma^{\le n}$ the set of all strings with size $\le n$. As domain of (all) compression algorithms, we consider the set of (all) *injective* functions in $\Sigma^\star \to \Sigma^\star$, i.e., functions that map any input string to some output string (encoding), where no two strings map to the same output.

Solutions for Problem 3 (No Free Lunch)

1. Let $\Sigma = \{0,1\}$. Assume, A is a compression method that always reduces its input size. That means, $A(\Sigma^n) \subseteq \Sigma^{\le n-1}$. But we have $|\Sigma^n| = 2^n$ whereas

$$|\Sigma^{\le n-1}| = \sum_{i=0}^{n-1} 2^i = 2^n - 1,$$

so A cannot be injective, a contradiction.

An alternative argument is by applying A iteratively. If A would always reduce the input size, after at most n steps, we have $A(A(\cdots(w))\cdots) = \varepsilon$ the empty string, for any input $w \in \Sigma^n$. Since A has a unique inverse A^{-1}, its decoder, applying A^{-1} some $k \le n$ times to ε must reproduce every $w \in \Sigma^n$, but obviously $(A^{-1})^k(\varepsilon)$ can produce at most n different source texts (for $k = 1, \ldots n$) in Σ^n, whereas $|\Sigma^n| = 2^n > n$ for $n \ge 2$.

2. We note that

$$|\Sigma^{\le n}| = \sum_{i=0}^{n} 2^i = 2^{n+1} - 1\ .$$

Consider $A(\Sigma^{\le n})$, i.e., the set of codewords assigned to all strings up to length n. These are $2^{n+1}-1$ many strings, but there are only $|\Sigma^{\le n-1}| = 2^n - 1$ bit strings that are *strictly* shorter than n. That means $A(\Sigma^{\le n})$ has to contain $2^{n+1}-1-(2^n-1) = 2^n$ strings w of length at least n; for any of these holds $A(w) \ge |w|$, and they comprise more than half of $\Sigma^{\le n}$.