Tutorial 7 Exercise Solutions

COMP526: Efficient Algorithms

18–19 November, 2024

Exercise 1. Compress the text T = `HANNAHBANSBANANASMAN` using a Huffman code; give

1. the character frequencies,
2. a step-by-step construction of the Huffman tree,
3. the Huffman code, and
4. the encoded text.
5. Finally, compute the compression ratio of the result (ignoring space needed to store the Huffman code).

Recall that we use the following conventions for building a Huffman tree:

- When merging two characters/nodes in the tree, the lower weight node becomes the left (0) child of the parent.
- Whenever there is a tie between weights, the node containing the alphabetically first child becomes the the left child of the parent.

Solution.

1. Frequencies:

char	A	B	H	M	N	S
weight	7	2	2	1	6	2

2. We proceed step by step; remember the tie-breaking rules!

   ***Note:** Using the correct tie-breaking rules is absolutely critical!*
   If there were ambiguity in the construction of the code, an encoder-decoder pair might not be able to correct reconstruct the source text.

   (i) mins: `M` and `B`, new symbol $\{\boxed{\text{MB}}\}$ with weight 3.

char	A	$\{\boxed{\text{MB}}\}$	H	N	S
weight	7	3	2	6	2

   (ii) mins: `H` and `S`, new symbol $\{\boxed{\text{HS}}\}$ with weight 4.

char	A	$\{\boxed{\text{MB}}\}$	$\{\boxed{\text{HS}}\}$	N
weight	7	3	4	6

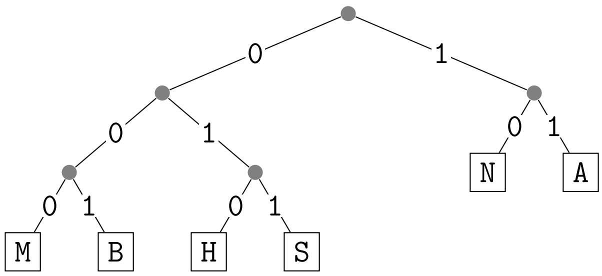
   (iii) mins: $\{\boxed{\text{MB}}\}$ and $\{\boxed{\text{HS}}\}$; new symbol with weight 7

char	`A`	$\left\{\boxed{\{\boxed{\text{MB}}\}\{\boxed{\text{HS}}\}}\right\}$	`N`
weight	7	7	6

(iv) min: `N` and `A`, new symbol $\{\boxed{\text{NA}}\}$ with weight 13

(v) only two left, new symbol $\left\{\boxed{\left\{\boxed{\{\boxed{\text{MB}}\}\{\boxed{\text{HS}}\}}\right\}\{\boxed{\text{NA}}\}}\right\}$

Note that the nested boxes encode the trie uniquely:



3. The actual Huffman code is

char	`A`	`B`	`H`	`M`	`N`	`S`
code	`11`	`001`	`010`	`000`	`10`	`011`

4. $C =$ `010 11 10 10 11 010 001 11 10 011 001 11 10 11 10 11 011 000 11 10`

5. Compression ratio is
$$\frac{47}{20 \cdot \lg(6)} \approx 0.909$$

□

Exercise 2. 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 $\left|\Sigma^{\leq n}\right| = \left|\Sigma^0\right| + \left|\Sigma^1\right| + \cdots + |\Sigma^n|$.

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

We denote by $\Sigma^\star$ the set of all (finite) strings over alphabet Σ and by $\Sigma^{\leq n}$ the set of all strings with size $\leq n$. As the 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.

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

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

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 $k \leq n$ 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 $n \geq 2$.

2. We note that

$$\left|\Sigma^{\leq n}\right| = \sum_{i=0}^{n} 2^i = 2^{n+1} - 1\,.$$

Consider $A(\Sigma^{\leq n})$, 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 $|\Sigma^{\leq n-1}| = 2^n - 1$ bit strings that are *strictly* shorter than n. That means $A(\Sigma^{\leq n})$ has to contain $2^{n+1} - 1 - (2^n - 1) = 2^n$ strings w of length at least n; for any of these holds $A(w) \geq |w|$, and they comprise more than half of $\Sigma^{\leq n}$.

□