Homework 7

Instructor: Rotem Oshman
Name: Ishan Pranav

References

Peer-reviewed with Crystal Huang.

Question 1: Huffman codes

1. Consider a file that uses the following list of symbols with the corresponding frequencies:

Letter	A	B	C	D	E	F	G
Frequency	0.06	0.09	0.10	0.12	0.15	0.16	0.32

Find an optimal prefix code based on Huffman's algorithm (using the symbols 0 and 1 only). Work out the code by drawing the tree and then describing the mapping from symbols to bit strings.

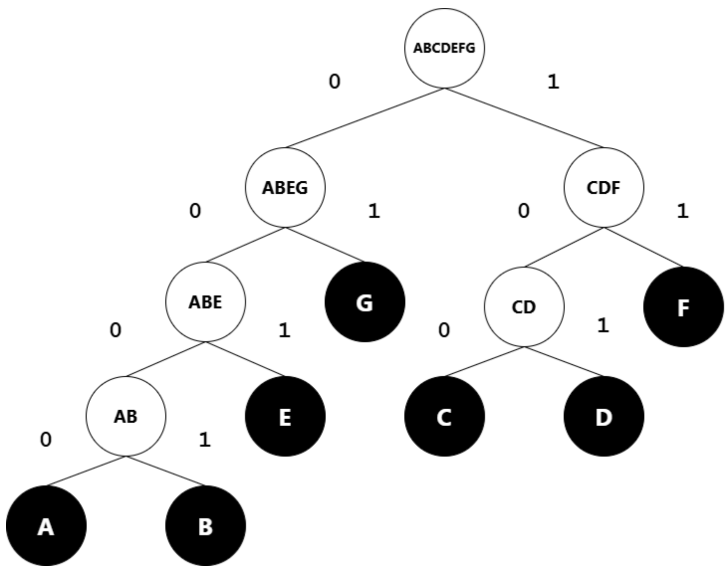

Figure 1: A Huffman tree for the given list of symbols and their frequencies, constructed using Huffman's algorithm.

Solution: We can step through the recursive levels of Huffman's algorithm to construct a Huffman tree and obtain an optimal prefix code.

First. The two lowest-frequency symbols are A and B with $f_{\mathrm{A}} = 0.06$ and $f_{\mathrm{B}} = 0.09$, respectively. We replace them with the pseudo-symbol AB such that $f_{\mathrm{AB}} = 0.06 + 0.09 = 0.15$.

Second. The two lowest-frequency symbols are C and D with $f_{\mathrm{C}} = 0.10$ and $f_{\mathrm{D}} = 0.12$, respectively. We replace them with the pseudo-symbol CD such that $f_{\mathrm{CD}} = 0.10 + 0.12 = 0.22$.

Third. The two lowest-frequency symbols are AB and E with $f_{\mathrm{AB}} = f_{\mathrm{E}} = 0.15$. We replace them with the pseudo-symbol ABE such that $f_{\mathrm{ABE}} = 0.15 + 0.15 = 0.30$.

Fourth. The two lowest-frequency symbols are F and CD with $f_{\mathrm{F}} = 0.16$ and $f_{\mathrm{CD}} = 0.22$. We replace them with the pseudo-symbol CDF such that $f_{\mathrm{CDF}} = 0.16 + 0.22 = 0.38$.

Fifth. The two lowest-frequency symbols are ABE and G with $f_{\mathrm{ABE}} = 0.30$ and $f_{\mathrm{G}} = 0.32$. We replace them with the pseudo-symbol ABEG such that $f_{\mathrm{ABEG}} = 0.30 + 0.32 = 0.62$.

Sixth. The two remaining symbols are ABEG and CDF. We construct parent vertex ABCDEFG and take ABEG and CDF to be children of ABCDEFG.

Then, we reconstruct the tree by unwinding the recursion and re-attaching the terminal symbols, taking the psuedo-symbols to be their parent vertices. Finally, we can arbitrarily assign all left edges to represent a 0 bit and all right edges to represent a 1 bit. See Figure 1.

We can now construct the optimal prefix code for each symbol by traversing the tree from root to leaf and taking note of the edges followed:

Letter	Encoding
A	0000
B	0001
C	100
D	101
E	001
F	11
G	01

This completes our encoding.

2. In general, are Huffman codes unique? That is, for a given set of letters and corresponding frequencies is there a unique Huffman encoding? Note that different letters may have the same frequency in the general case. If yes, justify. Otherwise provide a counterexample.

Solution: No, in general, Huffman codes are not unique.

First, bits are assigned to edges arbitrarily. For a given parent vertex, it is possible to let its left edge represent the 0 bit and the right edge represent the 1 bit, or to let the right represent 1 and the left represent 0. The only constraint on the assignment of bits is that one side must represent 0 and the other must represent 1.

Furthermore, we can show that Huffman codes are not unique even with a consistent constraint on the assignment of the bits.

Without loss of generality, assume that for every parent vertex, we always let its left child represent the 0 bit and the right child represent the 1 bit.

Consider three symbols A, B, and C, with frequencies $f_{\text{A}} = f_{\text{B}} = f_{\text{C}} = 0.\overline{3}$. We want to construct an optimal prefix code using Huffman's algorithm. On the first recursive level, we may choose any of A and B, B and C, or C and A to be the two lowest-frequency symbols. Consider two of these cases:

- Suppose we take A and B to be the two lowest-frequency symbols. Then we replace them with parent AB with frequency $f_{\text{AB}} = 0.\overline{6}$, with A as the left child and B as the right. Now AB and C are joined to a parent to construct the tree, with AB as the left child and C as the right.

  In our optimal Huffman code, we represent A with 00, B with 01, and C with 1.

- Suppose instead we take B and C to be the two lowest-frequency symbols. Then we replace them with parent BC with frequency $f_{\text{BC}} = 0.\overline{6}$, with C as the left child and b as the right. Now A and BC are joined to a parent to construct the tree, with A as the left child and C as the right.

  In our optimal Huffman code, we represent A with 0, B with 10 and C with 11.

We have shown that with the same list of symbols and frequencies, Huffman's algorithm may produce entirely different optimal prefix codes. Thus, Huffman codes are not unique.

Question 2: Minimizing number of colors

Let X be a set of n intervals $[s_i, f_i)$ on the real line. A *proper coloring* of X assigns a color to each interval so that overlapping intervals are always assigned different colors. In this problem, we want to find a proper coloring of X using a minimum possible number of colors. Let us call this (unknown) number c^*.

1. Given a problem instance $\{[s_i, f_i)\}$ and any point on the real line t, let n_t denote the number of sets that contain t. Also, let $c = \max_t n_t$ (easy to see the maximum is well defined). Argue that $c^* \geq c$.

Solution: Proposition I. *Claim.* Let X be a set of n real intervals $[s_i, f_i)$ for $1 \leq i \leq n$. For all $t \in \mathbb{R}$, let n_t denote the number of intervals $x \in X$ where $t \in x$. Let $c = \max\limits_{t \in \mathbb{R}}(n_t)$. Let c^* be the minimum number of colors required to produce a proper of coloring of X. Then $c^* \geq c$.

Proof. Since there exists $c = \max\limits_{t \in \mathbb{R}}(n_t)$, there exists $t^* \in \mathbb{R}$ where $c = n_{t^*}$.

By definition, there are n_{t^*} intervals $x \in X$ where $t^* \in x$. So, n_{t^*} intervals overlap at the point t^*. In a proper coloring of X, each of these n_{t^*} overlapping intervals is assigned a unique color. This implies that there are at least n_{t^*} colors in a proper coloring of X. Since c^* is the minimum number of colors required to produce a proper coloring of X, we know $c^* \geq n_{t^*}$.

Of course, $c = n_{t^*}$. Ergo $c^* \geq c$. $\square$

2. Consider the following greedy algorithm for solving this problem.

    Sort the intervals according to s_i.
    Initialize the current number z of defined colors to 0.
    for $i = 1$ to n, perform the following greedy step.
        if next interval $a_i = [s_i, f_i)$ cannot be legally colored with any color $1 \leq j \leq z$
        then increment z by 1 and assign a_i the new color z.
        else color a_i with the smallest "legal" color $j \in \{1, \dots, z\}$

    Argue that the greedy algorithm above computes a valid coloring, and yet *never uses more than c colors*, where c is the quantity from part (a).

Solution:

Algorithm I. GreedyAlgorithm(X), where X is a set of n real intervals $[s_i, f_i)$ for $1 \leq i \leq n$.

For all $t \in \mathbb{R}$, let n_t denote the number of intervals $x \in X$ where $t \in x$. Let $c = \max_{t \in \mathbb{R}}(n_t)$.

Let $1 \leq j \leq (n+1)$ where $j = 1$ denotes the iteration before the loop begins, $j = i$ denotes the iteration before considering interval $[s_i, f_i) \in X$ for $i \leq n$, and $j = n+1$ denotes the step immediately after the loop terminates.

Let z_j denote the number of unique colors used at the beginning of iteration j.

Invariant I. *Claim.* For every step of the loop $1 \leq j \leq (n+1)$, we claim that

- there exist no $1 \leq (p \neq q) < j \leq n$ where $[s_p, f_p), [s_q, f_q) \in X$ are overlapping intervals with the same color;
- $z_j \leq c_j$ where $c_j = \max_{t \in \mathbb{R}}(n_{j_t})$ and n_{j_t} denotes the number of intervals $[s_i, f_i) \in X$ with $i < j$; and,
- $z_j \geq z_i$ for all $i < j$.

Proof. We can prove this invariant by induction on j.

Basis. Consider $j = 1$. Note that we are considering the case before the first iteration of the loop.

Since $j = 1$, there are no intervals $[s_i, f_i) \in X$ where $i < 1$, as no interval has been considered.

It follows that $n_{1_t} = 0$ for all $t \in \mathbb{R}$, so $c_1 = 0$. From Algorithm I, before the loop begins, $z_1 = 0$. So $z_1 \leq c_1$.

Thus the loop invariant holds at initialization.

Hypothesis. Consider $j = k$ where $1 \leq k \leq n$. Assume that at the beginning of iteration k,

- there exist no $1 \leq (p \neq q) < k$ where $[s_p, f_p), [s_q, f_q) \in X$ are overlapping intervals with the same color;
- $z_k \leq c_k$; and,

- $z_k \geq z_i$ for all $i < k$.

Inductive step. Consider $j = k + 1$. Note that this is the iteration immediately after considering interval k and before considering interval $k + 1$. There are two cases.

- Suppose $[s_k, f_k)$ cannot be legally colored with any color $1 \leq y \leq z_k$. This implies that for all $1 \leq y \leq z_k$, there exists an interval with color y that overlaps with $[s_k, f_k)$. That is, there exists some $t^* \in [s_k, f_k)$ where for all $1 \leq y \leq z_k$, there is an $x \in X$ with color y and $t^* \in x$.

  In this case, Algorithm I assigns $[s_k, f_k)$ the incremented color $z_{k+1} = z_k + 1$.

  Since t^* exists in at least one interval for each color $1 \leq y \leq z_k$, we know that t^* is in at least z_k-many intervals. Also, $t^* \in [s_k, f_k)$. Therefore t^* is in at least $z_k + 1 = z_{k+1}$ intervals; that is, $n_{k+1_{t^*}} = z_{k+1}$.

  From the second statement of the inductive hypothesis, $z_k \leq c_k$. Note that if $n_{k+1_{t^*}} > c_k$, then $c_{k+1} = n_{k+1_{t^*}}$. Otherwise, $c_{k+1} = c_k$. Regardless, $z_{k+1} \leq c_{k+1}$.

  From the third statement of the inductive hypothesis, $z_k \geq z_i$ for all $i < k$. Since $z_{k+1} = z_k + 1$, we know $z_{k+1} > z_k \geq z_i$ for all $i < k$, maintaining this property.

  The color z_{k+1} assigned to $[s_k, f_k)$ is strictly greater than, and thus different from, all other colors used thus far. There is no previously-visited interval overlapping with $[s_k, f_k)$ with the same color z_{k+1}.

- Suppose instead $[s_k, f_k)$ can be legally colored with some color $1 \leq y \leq z_k$.

  Since $[s_k, f_k)$ can be legally colored with y, Algorithm I assigns the color y to this interval, respecting the first statement of the claim.

  In this case, Algorithm I does not increment the color. We know $z_{k+1} = z_k$, so the second and third statements of inductive hypothesis guarantee that these two properties of z_k are maintained for z_{k+1}.

In all cases, all three statements of the loop invariant hold.

Hence, by the principle of mathematical induction, the loop invariant holds for all iterations $1 \leq j \leq (n + 1)$. The loop invariant holds at initialization, maintenance, and termination.

Proposition II. *Claim.* Algorithm I produces a proper coloring using no more than c colors.

Proof. Algorithm I terminates when its loop terminates; that is, it terminates when $i = n$ on iteration $j = n + 1$.

From Invariant I, at termination there is no pair of distinct overlapping intervals with the same color. Therefore, Algorithm I computes a proper coloring of X.

At termination, the number of colors used is $z = z_{n+1}$, and $c = c_{n+1}$. From Invariant I, $z_{n+1} \leq c_{n+1}$. Therefore, Algorithm I uses $z \leq c$ unique colors.

Ergo, $\textsc{GreedyAlgorithm}(X)$ produces a proper coloring of X using $z \leq c$ unique colors. $\square$

3. What is the running time of this algorithm as a function of n and c^*?

Solution: **Proposition III.** *Claim.* Let c^* be the minimum number of colors required to produce a proper of coloring of X. Then $\textsc{GreedyAlgorithm}(X)$ has running time $O(n \log n + nc^*)$.

In the worst case, all intervals overlap, so $c^* = n$ and the running time is $O(n^2)$.

Proof. First, $\textsc{GreedyAlgorithm}(X)$ sorts the n intervals in X. This is an $O(n \log n)$ process.

Then, for each of the n intervals in X, the algorithm performs the main loop. In the worst case, the loop body considers all z previously-defined colors from 1 to z before determining that no such color is legal for the current interval. Of course, $z = c$ in the worst case. This is an $O(c)$ process. The loop body always branches, sometimes increments z, and always assigns z to the current interval: these execute in $O(1)$ time. So the loop body is an $O(c)+O(1) = O(c)$ process. Thus, the worst-case running time of the loop is $O(nc)$.

Therefore, the running time of Algorithm I is $O(n \log n) + O(nc) = O(n \log n + nc)$.

From Proposition I, we have $c^* \geq c$. Ergo Algorithm I has running time $O(n \log n + nc^*)$. $\square$