1 Prefix Codes as Trees

We first need to come up with a framework for designing prefix codes.

- **Idea**: represent a code as a binary tree.
- Each edge of tree is labeled with a bit (0 or 1).
- Left edges get 0’s, right edges get 1’s.
- Each letter \( x_i \) labels one leaf \( \ell_i \) of tree.
- Codeword corresponding to \( x_i \) is given by the bitstring labeling path from root down to \( \ell_i \).
- **Example**:

A few important observations...

- **Fact**: no two leaves get the same codeword (they have different paths from root).
- **Fact**: because letters appear only at leaves, code corresponding to tree is a prefix code.
- (Otherwise, some codeword would end at an internal node.)
- **Fact**: in tree for an optimal code (min total # of bits), every internal node has two children.
• **Pf:** Let $R$ be tree corresponding to a code, and suppose some int node $v \in R$ has one child $w$.

• Consider revised tree $R'$ that deletes edge $(v, w)$ and hangs subtree rooted at $w$ off of $v$.

• Codewords for all letters below $w$ in $R$ are one bit shorter in $R'$, and they do not collide with any other codewords from $R$. Hence, $R$ does not yield optimal code.

One more important definition.

• Let depth of leaf $\ell_i$, denoted $d(\ell_i)$, be length of codeword labeling path from root to $\ell_i$ in $R$.

• So, how many bits are used to represent text $T$ with tree $R$?

• All copies of letter $x_i$ together use $f_i \cdot d(\ell_i)$ bits.

• Hence,

$$B(T) = \sum_i f_i \cdot d(\ell_i).$$

2 Finding an Optimal Tree

We’ve reduced the problem to searching the space of all binary trees with $n$ leaves labeled with letters $x_i$, with 2 children per internal node. Goal is to find one labeled tree that minimizes $B(T)$.

• Intuitively, we want a tree that puts rare letters at high depth and common letters at low depth.

• **Idea:** build tree from bottom up. We will stick together subtrees until we have one full tree.

• Let $L = \{\ell_1, \ldots, \ell_n\}$ be set of leaves for all chars. Let $f_i$ be frequency of letter $x_i$ corresponding to leaf $\ell_i$.

• Find the two leaves $\ell_a$ and $\ell_b$ in $L$ with two lowest frequencies $f_a$ and $f_b$.

• Link these leaves into a single subtree $R_{ab}$, and create a new “megaleaf” $\ell_{ab}$ with frequency $f_a + f_b$.

• Recursively solve problem on reduced input

$$L \cup \{\ell_{ab}\} - \{\ell_a, \ell_b\}.$$

• Stop when $L$ contains one megaleaf for whole tree.
Finally, expand each megaleaf recursively from root to get final tree.

Note that this is a greedy algorithm: repeatedly join two least frequent leaves into one, until only one leaf remains.

3 Proof of Optimality

We take the usual route to an optimality proof for greedy algorithms.

- **Greedy Choice:** Let \( L \) be an instance of the encoding problem, and let \( \ell_a, \ell_b \in L \) be first two leaves chosen for linking by greedy algorithm. Then there exists an optimal tree for \( L \) containing \( R_{ab} \).

- **Pf:** Let \( R \) be any optimal tree for \( L \). If \( R_{ab} \) is subtree of \( R \), we are happy!

  - Otherwise, let \( \ell_x \) and \( \ell_y \) be a pair of leaves in \( R \) with common parent, such that \( \delta = d(\ell_x) = d(\ell_y) \) is maximal.

  - Assume that neither \( x \) nor \( y \) is one of \( a, b \). (If it is, simplified version of following argument still works.)

  - Modify \( R \) to obtain a new tree \( R^* \) by exchanging the positions of \( \ell_a \) with \( \ell_x \) and \( \ell_b \) with \( \ell_y \). Now \( R^* \) contains \( R_{ab} \).

  - Let \( B_R(T) \) be number of bits used for text \( T \) by \( R \)'s encoding.

  - For new tree \( R^* \), number of bits is given by

    \[
    B_{R^*}(T) = B_R(T) - (f_x + f_y)\delta - f_a d(\ell_a) - f_b d(\ell_b) + (f_a + f_b)\delta + f_x d(\ell_a) + f_y d(\ell_b)
    \]

    \[
    = B_R(T) + (f_a - f_x)(\delta - d(\ell_a)) + (f_b - f_y)(\delta - d(\ell_b)).
    \]

  - Note that, by greedy choices of \( a \) and \( b \), \( (f_a - f_x) \leq 0 \) and \( (f_b - f_y) \leq 0 \).

  - Moreover, by choice of \( x \) and \( y \), \( (\delta - d(\ell_a)) \geq 0 \) and \( (\delta - d(\ell_b)) \geq 0 \).

  - Conclude that \( B_{R^*}(T) \leq B_R(T) \), and so \( R^* \) is optimal too. QED

One down, two to go.

- **Inductive Structure:** first step of greedy algorithm leaves us with smaller instance \( L' \) of same problem without external constraints.
• Pf: let $L'$ be set of leaves after first linking step. $L'$ contains a smaller set of leaves with associated frequencies. Any strategy for joining the leaves of $L'$ into a tree is compatible with our greedy choice, since we can replace the megaleaf $R_{ab}$ in the final tree by the subtree with leaves $\ell_a$ and $\ell_b$.

Two down, one to go.

• Optimal Substructure: let $R'$ be an optimal tree constructed from leaves of $L'$ after linking $\ell_a$ and $\ell_b$. Then $R$, the tree obtained by replacing $\ell_{ab}$ in $R'$ with subtree $R_{ab}$, is optimal.

• Pf: Let $d'(\ell_{ab})$ be depth of megaleaf $\ell_{ab}$ in $R'$. Then in $R$, we have

$$d(\ell_a) = d(\ell_b) = d'(\ell_{ab}) + 1.$$ 

• Conclude that $B_R(T)$ is given by

$$B_R(T) = B_{R'}(T) - (f_a + f_b)d'(\ell_{ab}) + f_a d(\ell_a) + f_b d(\ell_b)$$

$$= B_{R'}(T) - (f_a + f_b)d'(\ell_{ab}) + (f_a + f_b)(d'(\ell_{ab}) + 1)$$

$$= B_{R'}(T) + f_a + f_b.$$ 

• Does usual contradiction argument work? Yes!

• (Exercise: work through it.) QED

4 Complexity

How efficiently can we implement Huffman coding?

• Maintain leaf set $L$ as priority queue keyed on frequency.

• To find two least frequent leaves in $L$, do two extractMin ops.

• Then insert new megaleaf back into $L$.

What does this cost?

• Each linking phase does two extractions and one insertion.

• Each of these three ops is $O(\log n)$ w/binary heap.

• There are only $n - 1$ phases, so total cost is $O(n \log n)$. 

4