Data Compression

We consider a basic problem in data compression, where the goal is to represent text in binary with as few bits as possible. In the approaches we will consider, a binary string is associated with each letter in the alphabet of the text, and the text is encoded simply by concatenating the codes for the letters in order. There are two types of codes:

- **Fixed-length encoding**: given k letters in an alphabet the ceiling of $\log k$ bits is used to represent each letter.
- **Variable length encoding**: the number of bits used to encode a letter depends on the frequency of the letter in the text.

	a	b	c	d	e	f	
frequency	45	13	12	16	9	5	(total = 100)
Fixed-Length	000	001	010	011	100	101	(300 bits)
Variable-Length	0	101	100	111	1101	1100	(224 bits)

The variable length encoding produces a 25% savings in file length. Both encoding methods are examples of a **prefix code** where no code word is a prefix of any other code word. **Decoding** the text is very straightforward: repeatedly read bits of the binary encoding until the code for some letter is read, and then output that letter.

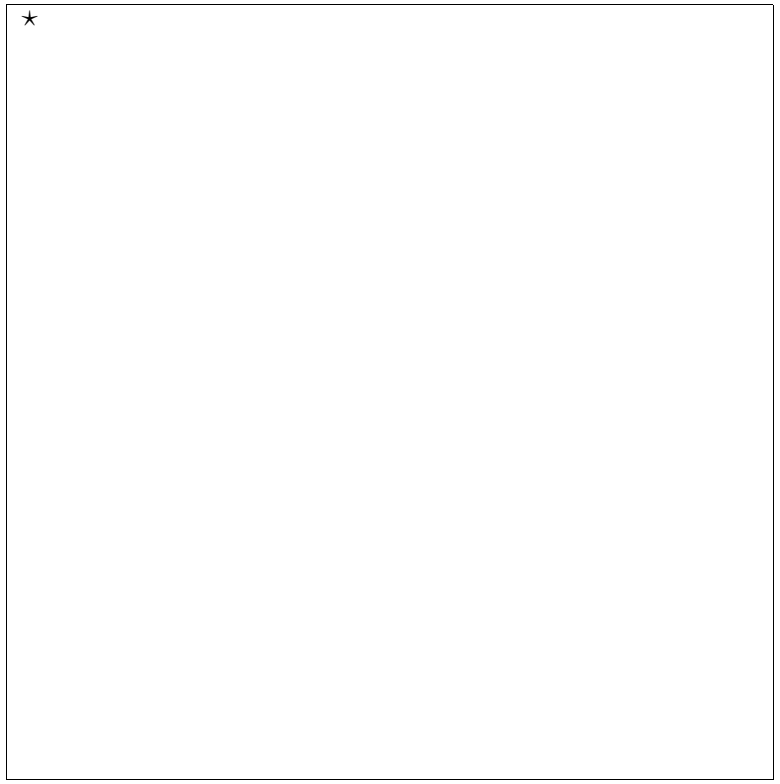
Our problem is to find an optimal prefix code for a given text, given that the frequencies of the letters are known. By an optimal prefix code, we mean a prefix code that minimizes the total number of bits needed to encode the file. More precisely, for each letter c, let $f(c)$ represent the frequency, the number of times c occurs in a file. Let $d_T(c)$ be the length of the codeword for c as given by prefix tree T. Then the number of bits needed to encode the file is

$$B(T) = \sum_c f(c) d_T(c)$$

where T represents the given prefix tree and c is in the file's alphabet.

Huffman's Algorithm

The following algorithm, due to Huffman, finds the optimal prefix tree for a given set of letter frequencies.



Proof of correctness

Claim 7 *Let C be the alphabet, let x, y be the two letters of lowest frequency. Then, there is an optimal prefix code in which the code words for x and y have the same length and differ only in the last bit [i.e. x, y are siblings in the prefix tree].*

Proof: Let T be an optimal prefix tree. Let b, c be siblings of maximum depth in T. Assume $f(b) \leq f(c)$ and $f(x) \leq f(y)$. Now exchange x with b and y with c. Note that $f(b) \geq f(x)$ and $f(c) \geq f(y)$. Since we are moving letters with lower frequencies down the tree, the new tree T' obtained from these exchanges has cost no greater than T, i.e. $B(T') \leq B(T)$. Hence, T' is also optimal and in T' x and y are siblings.

Claim 8 *Suppose T' is an optimal prefix tree for $(C - \{x, y\}) \cup \{z\}$ where z has frequency $f(x) + f(y)$ and x, y are letters with lowest frequency in C. Then the tree T obtained from T' by making x and y children of z is an optimal prefix tree for C.*

Proof: Let S be any prefix tree for C. We need to show that $B(T) \leq B(S)$. By Claim 7, we need only consider prefix trees S in which letters x and y are siblings. For such a tree S, let S' be the tree obtained by removing x, y from tree S. Clearly S' is a prefix tree for $(C - \{x, y\}) \cup \{z\}$. Therefore, $B(S') \geq B(T')$.

Also, comparing S and S', we see that

$$B(S') + f(x) + f(y) = B(S).$$

Similarly,

$$B(T') + f(x) + f(y) = B(T).$$

Putting these tree identities together, it follows that $B(T) \leq B(S)$, as desired.