

Applications Homework 01 (due Friday, May 12)

Directions: Write up carefully argued solutions to the following problems. Your solution should be clear enough that it should explain to someone who does not already understand the answer why it works. You may use results from lecture and previous homeworks without proof.

0. Decompressing Huffman Codes (25 points)

Retrieve the original text by writing a program to decode the following text compressed using Huffman Codes:

dict: {"00"=32, "01000"=100, "010010"=109, "010011"=119, "0101"=97, "01100"=46, "011010"=44, "01101100"=84, "011011010"=83, "011011011"=48, "0110111"=73, "0111"=111, "10000"=104, "10001"=115, "10010"=110, "1001100"=33, "100110100"=76, "100110101"=122, "10011011"=65, "100111"=99, "10100"=108, "10101"=105, "1011"=101, "11000"=114, "110010000"=79, "1100100010"=113, "1100100011"=78, "11001001"=72, "11001010"=63, "11001011000"=56, "11001011001"=68, "11001011010"=69, "11001011011"=85, "11001011100"=86, "11001011101"=120, "11001011110"=70, "11001011111"=77, "1100110"=39, "1100111"=112, "110100"=103, "11010100"=89, "1101010100"=49, "1101010101"=106, "110101011"=87, "1101011"=98, "110110"=121, "1101110"=107, "1101111"=102, "1110"=10, "11110"=116, "111110"=117, "11111100"=118, "11111101"=66, "1111111"=45}

msg: 9b9e77c22b2d0f38b4a1ba4e9c8a2a71e7de2fe557d57935dc1de85e2cac49389aec29aeefa307fa8835d8bae9679cdf4d99dc1cdfd113acb488b8b3cee4525a50f39a5fbabe89bd7c52bef52cd6e8d87dfbcf42cd30ff490cee0e6c859aeeda1f793bfd88db46fa56e22cb64d76ee0ed773afd1b35ddc443cb3789d717c9c17c43e496513d0acb7158954b3bc635d74b677076a5d28e9b46ba2cfb986a5d28e9b46ba2cfb99da974a3a6d1ae8b3ee61a974a3a6d1ae8b3ee67707641e0d1ae8b3ee164836ba51d399d34bf6689182eeb2be3ece294afbd4b677077eae31b4c3f716bbb7ac7c56262d51b4cee0ec83a5596933b83b2565a0f2148dcef2433b83b26e947caee0efe7eae31b6577f26d0a96577077f320b2367f8d774adfe59e85625624173e7b9565a6577f1b93acb379837cd4a19eb3ee367f8fb3b3b83a68efbab2d08c1719d9dc1d96e367a1647cb5c4586a3fd837afa178eceb541a3ba049e57b1bdf07a0f1b67707369f18d9837cc9179767afad0cee0ec9bc5e689e859a6151f2fad9dabbcd8b3f2f1b19f0ff207de6cff3445e4cee0e9b33de37dcfe3179df1e275c21a16943f73459dc1d972f1b19f0ff219dc1d97d660dcd1fc29e8565a1a3d65a10bc5b3b83bf9a8ff1a3f8a565e1e46cff60dff4d66acefe64139836417d9a225733b83bf9ab5fcb3ce7d1986aef35286bb2b230e99aa6a9961a6db6db6cfe7eed733b83bf5718d9a1b9e7a20d9fe6ba3e7ce6fa6d59685647a1641fe8dccee0efe64decdd13685499dfcc9bd9a3f5718d99dc1dfc6f13d05f1bfe9acd4d2e995dfc9b295f7a96c1b59ee7aad042ecd1a6151f2faaf2677076477f9783d07fb48786f99024bbacaf99dc1cd90c5d885da2698545919e0ef46bbd0c5d885da242ba408cf077a33b839b20f1b13bc58a9ee4d7086770736431762176893bd297a5b06f99234a2a06e79df772885d85920857d3c215dd68170ff4907a1642bf96cee0ed47f845504ee959ad67dc22bbf7de2e5e677077f326ad1fab8c6ccefef4de3d5b68d30e9d65a0a25f47cb3c2f289a3bee89a3ba19dc1dfcc9b5c0badfef197b0b2dd5bcaefe6a5ac19dc1dfcd47f8d1eb2d0f39e859bfe95e16995dfcc8dda1be64893f8d1eb2d19dc1d96bf2f1b6b746c6e93a716ba3eacb422e95bcada367f8456d9dc1d965e59c26b1f595f0f2148e45f5718ba339b5f4f01483fa16a8677073734fae312164eff510417f2fb55fa3e347f7ae43fde1f79e8589aeccee0e6e51ff2cf42b12b29df19df0beacb4e590a97d1b4ae5e33ae3715979c7fb021766770736417d9a2270d84ec43cb37895da0fc59d7d579459dc1dfafb34645f57d629aebef1f799dd2cec633be922f07a164f5c4ff251f2ca7b8b3b83bf9abba51a26d0a93479d0bb09d8b8b12b933bf9abb171733b83bf9fb77fd2b933bf93695ca6770764ad2974fb55584cee

1. Lempel-Ziv: A Whole New Game (75 points)

Lempel-Ziv is a family of compression algorithms that attempt to compress a string by finding repeated “phrases”. For example, aaaaaaaaaaaaaa and abcabcabcabc should be highly compressible.

In this problem, we will deal with a slightly simplified LZ algorithm which splits a string into phrases greedily. The algorithm repeatedly chunks the input into phrases by splitting at the smallest prefix it hasn’t already seen. For example, for the string AABABBBABAABABBBABBABB, LZ would first find A, then AB, then ABB, etc. This would result in a string split into phrases as follows:

A | AB | ABB | B | ABA | ABAB | BB | ABBA | BB

As we split the string, we number the phrases we’ve found: $\{A = 1, AB = 2, \dots\}$. Then, to output a compressed version of the string we replace the repeats with their numbers:

A | AB | ABB | B | ABA | ABAB | BB | ABBA | BB
A | 1B | 2B | B | 2A | 5B | 4B | 3A | 7

- (a) [10 Points] Explain in your own words why combining Huffman Coding with LZ might lead to better compression. In what way might you combine them?

A Phrase Repeated

We call an individual code that LZ outputs a “phrase”.

Define S^n , where S is some string and the power indicates “repeat n times”. So for example, if $S = \text{foo}$, then $S^3 = \text{foofoofoo}$.

- (b) [10 Points] Imagine we compress S^n using LZ for some S of non-zero length. Find a lower bound, in terms of n , on how many “phrases” the output would be asymptotically.

Compression Cannot Be Perfect

- (c) [10 Points] Prove that it is not possible to compress all strings of length n , on an arbitrary alphabet Σ . That is, prove that no compression scheme can make **every** string shrink in length.

LZ Worst-Case

By the previous part, it follows that we should consider a compression scheme to be “good” if the worst case string doesn’t expand “too much”. In other words, since some strings have to grow for others to shrink, we want to find a limit to how bad the most expanded string is.

Suppose, for simplicity, that $\Sigma = \{0, 1\}$.

Throughout, you may assume an explicit finite lower bound (like 3 or 10, or even 1000 if you want) on k (defined below), if it helps.

- (d) [5 Points] Consider the set of strings where the length of the largest phrase generated by LZ is k . Find the string, S_k , in this set that maximizes the number of phrases LZ will use.
- (e) [5 Points] Prove that $|S_k| = (k - 1)2^{k+1} + 2$.
- (f) [5 Points] Let $c(X)$ be the number of phrases X is parsed into. Show that $c(S_k) = 2^{k+1} - 1$.
- (g) [5 Points] Prove that $c(S_k) \leq \frac{|S_k|}{k - 1}$ using the two previous parts. You may assume $k > 1$.

Let Q be an arbitrary string of length n . Let k be the largest x such that $|S_x| \leq n$.

- (h) [5 Points] The maximum possible value of $c(Q)$ will occur when Q starts with S_k and finishes with some string A to pad to n . Let this worst case string be W . Prove that $c(A) \leq \frac{|A|}{k+1}$.
- (i) [10 Points] Show that $\lg c(Q) \leq k + 2$. Use this together with the previous two parts to prove that $c(Q) \leq \frac{n}{\lg(c(Q)) - 3}$.
- (j) [5 Points] Explain why the largest number of bits Q can be compressed to by LZ is $c(Q) \lg(c(Q)) + c(Q)$.
- (k) [5 Points] Finally, conclude that the largest number of bits that Q can be compressed into is $n + \ell \frac{n}{\lg(n)}$ for some natural number ℓ .