Problem 1: Move-to-Front Decoding

The move-to-front heuristic is a simple compression heuristic that can take advantage of patterns in data. The idea is to keep the symbols to be compressed ordered. Whenever you read a new symbol to compress, output its position in the ordered symbols and then move it to the front. For example if we start with symbols ordered as \{A, B, C, D, E\}, with A at the front, the string BBCAEB would be converted to \{1, 0, 2, 2, 4, 3\}. After generating these numbers they can be compressed using Huffman or arithmetic coding (we assume we can see the whole sequence before compressing it). The hope is that there are many small numbers due to "temporal locality" of the symbols so the compression will work well.

In the following questions assume we are using the move-to-front heuristic followed by arithmetic coding.

(A) What do we have to send in order to uncompress the data. (Answer in no more than two sentences.)

(B) If we had a long string with the substring ABCAB repeated over and over, asymptotically how many bits-per-symbol would the move-to-front heuristic use to store the string.

(C) If we had a long string with the substring ABCDE repeated over and over, asymptotically how many bits-per-symbol would the move-to-front heuristic use to store the string.

(D) Would the move-to-front heuristic adapt well to changes in types of text within a single “document” (e.g. compressing a tar file containing a mix of English, C and postscript files).

(E) Would coding pairs of symbols (i.e. keep all possible pairs ordered, and code two symbols at a time) help get better compression for long messages? Assume we are compressing the English language.

Problem 2: A variant on LDPC codes

Consider the following variant on LDPC codes. Like LDPC codes the code is given by a bipartite graph, but now assume that the neighbors for each node on the right must form a proper Hamming code. To be concrete lets assume each vertex on the right has degree 15 and the bits on the neighbors must form a (15, 11, 3) Hamming code. We will assume each vertex on the left has degree \(d = 3\) so the number of nodes on the right is \(n/5\).

1. What is the rate of this code (i.e. \(k/n\))? 

2. Assuming the bipartite graph has expansion \((\alpha, \beta)\) with \(\beta = d/2 = 1.5\) prove that the code has distance at least \(\alpha\). This is a similar argument to the one given in class for LDPC codes, but for LDPC codes we required that \(\beta > d/2\).
Problem 3: Burrows Wheeler

Decode the following bit string sent by Burrows Wheeler, assuming the integers generated by the move-to-front transformation are encoded using the gamma code (as defined on page 26 of the lecture 1 and 2 slides from the compression lectures). Assume that the string only has three characters and the original order in move-to-front is [a, b, c]—that is a is at location 1 (the front), b at location 2, and c at location 3 in the order. The last integer in the sequence of gamma codes specifies the position of the start character of the string.

1001011010010001110001

(A) What is the integer sequence obtained after decoding the gamma codes?

(B) Give the character sequence after decoding using move-to-front.

(C) What is the output string?

(D) Describe how to use Suffix Trees to do the Burrows Wheeler transform in linear time.

Problem 4: Covering Strings

Given strings A and B, a minimum cover of A by B is a decomposition \( A = w_1w_2\cdots w_k \) where each \( w_i \) is a substring of B and \( k \) is minimum. Show how to compute a minimum cover (given that one exists) in \( O(|A| + |B|) \) time. (Hint: Use suffix trees; you do not need to know or use the details of their construction).

Problem 5: Multiple Alignment

Consider the edit distance among three strings. An alignment among three strings is one in which the strings are padded with blanks so that every column only contains the same character or a blank. The triple-min-edit-distance problem seeks to find the alignment with the least length. For example, given the strings

\[
\begin{align*}
C & \quad A & \quad G & \quad A & \quad G \\
A & \quad T & \quad A & \quad G & \quad A \\
C & \quad T & \quad G & \quad A & \quad G
\end{align*}
\]

an optimal alignment is

\[
\begin{align*}
C & \quad A & \quad - & \quad - & \quad G & \quad A & \quad G \\
- & \quad A & \quad T & \quad A & \quad G & \quad A & \quad - \\
C & \quad - & \quad T & \quad - & \quad G & \quad A & \quad G
\end{align*}
\]

Give an algorithm to solve this problem. For three strings of length \( l, m \) and \( n \) it should take \( O(lmn) \) time. For extra credit solve the problem in \( O(\max(l, m, n)d^2) \) where \( d \) is the number of blanks that are added in the optimal solution.

Problem 6: Sketching the Stream

Given a stream of elements drawn from the universe \( D = \{1, 2, \ldots, N\} \), let \( f_i \) be the number of times the element \( i \) has been seen in the stream. Recall that \( S = \sum_{i \in D} f_i^2 \) is the second moment (the “self-join”) and you can compute it in space \( O(\log(N + \sum_i f_i)) \) using the algorithm of Alon, Mathias, and Szegedy. (See lecture slides.)

Now suppose you are given two disjoint streams (one on Earth, the other on the Moon). Both streams consist of elements from \( D \), and suppose \( (f_1, f_2, \ldots, f_N) \) and \( (f'_1, f'_2, \ldots, f'_N) \) are frequency vectors from the two streams. You want to calculate the second moment \( S'' \) of the union of the two streams, where \( S'' = \sum_{i}(f_i + f'_i)^2 \). Moreover, you want to do this using small space \( M = O(\log(N + \sum_i f_i + \sum_i f'_i)) \), and so that the amount of communication between the Earth and the Moon is only \( O(M) \). Give an algorithm for the problem.
Problem 7: Separators and Graph Embeddings

The bisection width of an $N$-node undirected graph is the size of the smallest set of edges whose removal partitions the graph into two disjoint sets of nodes, each with exactly $N/2$ nodes. In other words, the bisection width is the size of the smallest $\frac{1}{2} - \frac{1}{2}$ edge-separator.

One way to prove a lower bound on the bisection width of an $N$-node undirected graph $H$ is to first embed an $N$-node undirected graph $G$ with a known bisection width into $H$, and then reason from there.

1. What is the bisection width of the $N$-node complete graph $K_N$?
2. Suppose that each edge in the complete graph is replaced by a pair of edges, so that the graph is now a multigraph. What is the bisection width of this graph $2K_N$?
3. Embed $2K_N$ in the $d$-dimensional hypercube with $N = 2^d$ nodes where the dimensions are labeled 1 through $d$. For each pair of vertices $\{a, b\}$ in the complete graph, embed one path from $a$ to $b$ and one path from $b$ to $a$. Each path should be a shortest path, and should use the dimensions of the hypercube in non-decreasing order. What is the maximum congestion on any edge of the hypercube?
4. Suppose that the hypercube is bisected by removing a set of edges, and then for each hypercube edge that is removed, all of the edges in $2K_N$ whose paths use the hypercube edges are removed from $2K_N$. Prove that $2K_N$ is bisected as well.
5. Now prove a lower bound of $N/2$ on the bisection width of the hypercube.
6. Show that the lower bound is tight by exhibiting a bisection of the hypercube that contains $N/2$ edges.

Problem 8: Chernoff and Estimators

Suppose you are given a program that takes some random amount of time $T$ to execute, where $T$ is distributed according to some fixed but unknown distribution. The mean of this distribution is $E[T] = \tau$, and the variance is $E[(T - \tau)^2] = \sigma^2$. You succeed if you get an estimate for the mean running time which is in the interval $[(1 - \epsilon)\tau, (1 + \epsilon)\tau]$; you are happy to succeed with probability $1 - \delta$ (where $\delta \ll 1$).

For brevity, let us define $I_{\text{good}} = [(1 - \epsilon)\tau, (1 + \epsilon)\tau]$.

1. Suppose you run the program $n$ times independently, and get $n$ independent samples $T_1, T_2, \ldots, T_n$ for the running time. Note that the random variable $S_n = \frac{1}{n} \sum_{i=1}^n T_i$ has mean $\tau$. What is the variance of the random variable $S$? Use Chebyshev’s inequality to infer a bound on $\Pr[|S_n - \tau| \geq c\tau]$, and hence calculate the number of samples $n$ which would make this probability less than $\delta$. (Call this number $n_{\delta}$.)

2. Now we do something different: Let $m = n_{(1/4)}$ be the number of samples so that $\Pr[S_m \in I_{\text{good}}] \geq 3/4$. Note that you are succeeding with probability $3/4$, whereas you would like to succeed with probability $1 - \delta$. So you now create $k$ such independent sample-means $S_m^{(1)}, S_m^{(2)}, \ldots, S_m^{(k)}$ and take their median $M$.

   To analyze this, define the random variable $X_i = 1$ if the $i^{th}$ sample mean $S_m^{(i)}$ falls in the interval $I_{\text{good}}$, and $X_i = 0$ otherwise. What is the expectation $E[X_i]$? Use Chernoff bounds to give a lower bound on the number of indices $i \in \{1, 2, \ldots, k\}$ such that $X_i = 1$.

3. Use the result from the previous part to give an upper bound on the probability $\Pr[M \notin I_{\text{good}}]$. Hence show that setting $k = O(\log \frac{1}{\delta})$ will ensure that $\Pr[M \in I_{\text{good}}] \geq 1 - \delta$.

   How does the total number of samples taken compare to the naïve Chebyshev sampler of the first part?