Collaboration is encouraged, but students MUST write up solutions independently and list those with whom they worked. No materials from prior years’ classes or the Internet can be consulted.

1. (15 pts) Coming up with the Huffman codes for a set of letters \( C = \{c_0, c_1, \ldots, c_{n-1}\} \) is a fairly straightforward greedy algorithm. However, once the time comes to transmit a compressed text, you have to transmit the character/code pairs with it so that it can be decoded. Since Huffman codes can be quite long for infrequent characters, simply sending the pairs can take \( O(n^2) \) bits. To improve this, show how any set of Huffman codes on \( C \) can be represented by a sequence of \( 2^{n-1} + n \lceil \log n \rceil \) bits.

2. (24 pts: 4,4,8,8) A binomial list is a sorted list of \( 2^k \) items, for some integer \( k \geq 0 \). Store \( n \) items in a collection of binomial lists, such that there is a list of size \( 2^j \) for each \( n_j = 1 \) in the binary representation of \( n \). For example, if \( n = 13 = 1101_2 \), then the collection will have three sorted lists of the following sizes:

\[
\begin{align*}
\text{段1} & \quad \text{段2} & \quad \text{段3} \\
\text{段4} & \quad \text{段5} & \quad \text{段6} \\
\text{段7} & \quad \text{段8} & \quad \text{段9} \\
\text{段10} & \quad \text{段11} & \quad \text{段12} \\
\text{段13} & \quad \text{段14} & \quad \text{段15} \\
\text{段16} & \quad \text{段17} & \quad \text{段18} \\
\text{段19} & \quad \text{段20} & \quad \text{段21} \\
\end{align*}
\]

A convenient way to represent the collection of binomial lists uses a single array \( A[1..n] \) and stores the list for \( n_j = 1 \) in the segment \( A[(n-b)...(n-a)] \), where \( a = \sum_{i=0}^{j-1} n_i 2^i \) and \( b = (\sum_{i=0}^{j} n_i 2^i) - 1 \), as follows:

\[
\begin{align*}
\text{段1} & \quad \text{段2} & \quad \text{段3} \\
\text{段4} & \quad \text{段5} & \quad \text{段6} \\
\text{段7} & \quad \text{段8} & \quad \text{段9} \\
\text{段10} & \quad \text{段11} & \quad \text{段12} \\
\text{段13} & \quad \text{段14} & \quad \text{段15} \\
\text{段16} & \quad \text{段17} & \quad \text{段18} \\
\text{段19} & \quad \text{段20} & \quad \text{段21} \\
\end{align*}
\]

(a) \( A \) consists of a sequence of sorted segments. How many segments are there?
(b) Show how to search for a key in time \( O(\log^2 n) \).
(c) When you insert a new key, you may have to rearrange the lists so that they correspond to the 1-bits in the binary representation of \( n \). Show that this can be done in \( O(n) \) time in the worst case. (Recall that two sorted lists can be merged in time linear to the total number of their items).
(d) Start with an empty collection of binomial lists and insert \( n \) items in sequence. Show that the amortized time for an insertion is only \( O(\log n) \).

3. (24 pts: 8,8,8) Suppose that \( n \) keys are randomly inserted into an initially empty hash table with \( n \) slots that uses chaining. Each insertion is independent, and each incoming item is equally likely to end up in any list.

(a) What is the expected number of empty lists?
(b) What is the expected number of lists with exactly one key?
(c) What is the expected number of lists with \( \lg n \) keys?

4. (22 pts) Let \( X[1..n] \) and \( Y[1..n] \) be two arrays, each containing \( n \) numbers already in sorted order. Give an \( O(\lg n) \)-time algorithm to find the median of all \( 2n \) elements in arrays \( X \) and \( Y \).

5. (15 pts) Show that the second smallest of \( n \) elements can be found with \( n + \lfloor \lg n \rfloor - 2 \) comparisons in the worst case.