Substring Search

1. Brute force substring search copmares the search text to the search key character by character. The Rabin-Karp algorithm is very similar but instead hashes the length of the search key within the search text. We know that doing so will require $O(L)$ time. Under this scenario, how can the Rabin-Karp algorithm achieve better performance?

2. What is the difference between the Las Vegas version of the Rabin-Karp algorithm and the Monte Carlo version of the Rabin-Karp algorithm? Which one runs faster?

3. Anti-virus software can scan a binary file once and quickly determine whether any binary sequence matches known virus signatures. How can you modify the Knuth-Morris-Pratt algorithm to quickly and efficiently support the same operation?

4. Anti-virus software can scan a binary file once and quickly determine whether any binary sequence matches known virus signatures. How can you modify the Rabin-Karp algorithm to quickly and efficiently support the same operation? Which version of the Rabin-Karp algorithm would you use (Las Vegas or Monte Carlo)? Why?

5. Describe a modification to the Rabin-Karp algorithm to support searching for strings with wildcards in them. Wildcards can only match a single character (e.g. search for `going .here`, where the `.` can match any character).

Data Compression

6. Suppose that all symbol frequencies are equal. Describe the Huffman code.

7. Assume that we have a set of variable length codes: `[a: 0, b: 1001, c: 1011, d: 111, e: 1110]`. Is this set of variable length codes uniquely decodable? If not, find a string with two encodings.

8. Using Huffman coding, how many bits are needed to encode n copies of the symbol `a` (as a function of n)? How many bits are needed to encode n copies of the sequence `abc`?

9. Huffman coding relies on the frequency of each word in order to generate the optimal Huffman encoding trie. Suppose two words, w and x, have frequencies $f(w)$ and $f(x)$, respectively. If $f(w) > f(x)$ (the frequency of word w is greater than the frequency of word x), will the code word for w be shorter, longer, or equal to the code word for x? Will this always be the case?

10. Huffman coding could be extended in a straightforward way to encode in 2-bit characters (using 4-way trees). What would be the main advantage and the main disadvantage of doing so?