Problem Set 9

All parts are due on December 8, 2017 at 11:59PM. Please write your solutions in the LaTeX and Python templates provided. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct. Solutions should be submitted on the course website, and any code should be submitted for automated checking on alg.csail.mit.edu .

Problem 9-1. [20 points] **Programming Jumble**

Gill Bates wrote generic Python pseudocode to solve dynamic programming problems, using both top-down and bottom-up approaches. Unfortunately, the lines of his code got jumbled, and he needs your help to reconstruct his functions. Below are his two Python function definitions, as well as some lines of additional Python code. Help Gill by indicating the sequence of lines that completes each function. It may help to write out each function line by line, but a correct answer will simply be a sequence numbers corresponding to the lines that complete each function.

```
1.  else:
2.  j = i
3.  n = i
4.  i = j
5.  i = n
6.  return memo[n]
7.  for i in range(n):
8.  if i in base_cases:
9.  top_down_dp(i, memo)
10. top_down_dp(j, memo)
11. for i in dependencies(j):
12. for j in dependencies(i):
13. memo[i] = solve_base_case(i)
14. memo[i] = solve_from_subproblem_memo(i, memo)
```

Each line of code above may be used in both, only one, or neither function, and each function should be completed using only the lines provided (do not write additional code).

(a) [10 points]

```
def top_down_dp(n, memo = {}):
    # compute subproblems top down
```

(b) [10 points]

```
def bottom_up_dp(n, memo = {}):
    # compute subproblems bottom up
```

Problem 9-2. [40 points] **Coding Practice**

Alice wants to encode the message `yummybanana` using a small number of bits. Currently each letter is encoded using ASCII[1], a fixed-width coding scheme mapping characters to combinations of 7-bits. As such, Alice's message encoded in ASCII requires $7 \times 11 = 77$ bits to store. Since there are only six types of letters in her message, $\mathcal{X} = \{\texttt{a}, \texttt{b}, \texttt{m}, \texttt{n}, \texttt{u}, \texttt{y}\}$, she thinks she should be able to construct a lossless binary code to represent each letter using no more than 3 bits.

(a) [15 points] The Kraft inequality states that if character $x_i \in \mathcal{X}$ is represented using l_i bits in a uniquely decodable code, then $\sum_{x_i \in \mathcal{X}} 2^{-l_i} \leq 1$. Show there exists an assignment of bit lengths to letters in $\mathcal{X}$ that satisfies the Kraft inequality such that no code word uses more than 3 bits. For such a code, what is the maximum number of characters that can be assigned codewords using either one or two bits respectively?

(b) [10 points] Give a prefix-free code mapping each character in $\mathcal{X}$ to a codeword represented using exactly 3 bits. Draw a prefix tree associated with your code, with zeros branching left and ones branching right.

(c) [15 points] Your code from part (b) stores Alice's message using only $3 \times 11 = 33$ bits, a significant improvement! However, Alice heard that she might be able to compress her message even further by taking into account how often each character appears in her message. For each character in $\mathcal{X}$, calculate the frequency it appears in the string, i.e. (# occurrences / length of string). Then construct a binary Huffman code on the characters $\mathcal{X}$ using your frequencies, and draw its prefix tree. How many bits are needed to encode Alice's message using your Huffman code?

[1] `https://en.wikipedia.org/wiki/ASCII`

Problem 9-3. [20 points] **Expensive Ones**

Consider a source sequence of characters from a set $\mathcal{X}$, with each character $x \in \mathcal{X}$ occuring with probability $p(x)$.

(a) [10 points] Describe an algorithm to construct a binary prefix-free code that minimizes the expected number of ones in its encoding of the source; the bit lengths of codewords are not constrained.

(b) [10 points] Huffman's algorithm constructs an *optimal* binary prefix-free code, that minimizes the expected number of bits to encode the source. Modify Huffman's algorithm to return a code from among all optimal prefix-free codes that minimizes the expected number of ones in its encoding of the source.

Problem 9-4. [20 points] **Dependent Sources**

Consider the following five binary sources, each providing a temporally independent bit to you every second:

- X_1 is always 0;
- X_2 is randomly distributed, with a 1 occurring with probability 0.2;
- X_3 is randomly distributed, with a 1 occurring with probability 0.3;
- X_4 is the symbol-wise sum of X_1, X_2, and X_3 modulo 2; and
- X_5 is symbol-wise equal to either X_1 or X_2, each with probability 0.5.

You would like to compress one or more of these sources for future lossless recovery. Because some of these sources are not mutually independent, coding two of them together may provide higher compression than coding them separately, and Slepian-Wolf can be used to jointly encode them efficiently.

Recall that the **entropy** of a Bernoulli random variable X with $\Pr\{X = 1\} = p$ is given by $H(X) = -p \log_2 p - (1-p)\log_2(1-p)$, and that joint entropy is given by

$$H(A, B) = -\sum_{a \in A} \sum_{b \in B} \Pr\{A = a, B = b\} \log_2 \Pr\{A = a, B = b\} = H(A|B) + H(B).$$

(a) [5 points] State the individual entropy $H(X_i)$ for each source, for $i \in [1, 5]$.

(b) [15 points] State the joint entropy $H(X_i, \ldots, X_j)$ for the following five sets of sources, in terms of the individual source entropies $H(X_i)$. Which sets can theoretically be encoded at a lower bit-rate when encoded jointly rather than separately?

$$\{X_1, X_2\}, \{X_2, X_3\}, \{X_2, X_5\}, \{X_3, X_4\}, \{X_3, X_4, X_5\}$$