CSE 21 Fall 2022
Homework 5 ungraded problems

INSTRUCTIONS
These are problems you can discuss with classmates or people outside the class.
KEY CONCEPTS Inclusion/Exclusion, Ranking/Unranking, Huffman Encoding, Graph representations.
(Note: For this homework, you can leave your answers in terms of exponentials, factorials, binomial coefficients, etc.)

---

1. I am picking meals to offer at a catered party, from a list of 20 possible main dishes offered by a catering company. On that list 4 meals are vegan, 3 are gluten-free, and 2 are both (already counted in each category). How many sets of 3 options can I pick so that at least one is vegan and at least one is gluten-free?

   First, using inclusion-exclusion, since there are 4 vegan meals, 3 that are gluten-free, and 2 are both, there are a total of $4 + 3 - 2 = 5$ meals that are either vegan or gluten free. There are $\binom{20}{3}$ sets of 3 options. Let A be the set of sets of 3 options that have no vegan option, and B the set with no gluten-free option. Then $|A| = \binom{16}{3}$ since we must remove the 4 vegan meals before choosing 3, and $|B| = \binom{17}{3}$ for the same reason. $A \cap B$ is the set with neither a vegan nor a gluten-free option, so has size $\binom{15}{3}$ since we said that we need to remove 5 options. Using inclusion exclusion, $|A \cup B| = \binom{16}{3} + \binom{17}{3} - \binom{15}{3}$, and these are the options we want to exclude, so what's left is of size $\binom{20}{3} - \binom{17}{3} - \binom{16}{3} + \binom{15}{3}$

2. A 3-coloring sequence of length n (with $n \geq 1$) is a sequence of the colors red, blue and green (R,B,G) such that no color occurs in two consecutive entries.

   (a) How many 3-coloring sequences of length n are there? (give a brief justification.)

   There are 3 colors we can choose for the first position, but in each subsequent position, only 2 colors that are different from the previous one. Thus, there are $3 * 2^{n-1}$ possible colorings.

   (b) How many bits would the most efficient encoding of such a sequence use? (in terms of n.) (give a brief justification.)

   Using our general formula, there are $\lceil log_2(3 * 2^{n-1})\rceil = \lceil log_2 3 + (n - 1)\rceil = n + 1$ bits in the most efficient possible code.

   (c) Develop your own encoding/decoding algorithm where the code uses this number of bits. (Please give a brief description of how it works on an arbitrary input.)

Let's think of the three colors as being arranged around a 3-clock as $Blue, Green, Red$ in positions $0, 1, 2$. We can use the first two bits to express the corresponding position in binary for the first color. Then we can use 0 to mean that the next value is the one clockwise (+1 mod 3) and 1 to mean that the next value is the one counter-clockwise (-1 mod 3) We use 2 bits for the first value, and 1 bit for each of $n-1$ subsequent colors, for a total of $2+n-1 = n+1$ bits.

3. A *slow-growing sequence* of length n is a *non-decreasing* sequence of integers that start with 1 and each pair of entries differ by at most 1.

   Here are a few examples of *slow-growing sequences* of length 8:

   $(1, 1, 1, 1, 1, 1, 1, 1)$, $(1, 2, 2, 2, 3, 3, 4, 5)$, $(1, 2, 3, 4, 5, 6, 7, 8)$

   (a) (For $n \geq 1$), How many *slow-growing sequence* of length n are there? (give a brief justification.)

   The first value is forced, just 1. If we have value v_i in the i'th place, we can choose $v_{i+1} = v_i + 1$ or $v_{i+1} = v_i$ and those are the only choices. So there are 2 choices for each subsequent position. Thus there are 2^{n-1} slow-growing sequences total.

   (b) How many bits would the most efficient encoding of such sequences use? (in terms of n. Justify your work)

   The most efficient representation uses $log_2(2^{n-1}) = (n-1)$ bits.

   (c) Develop your own encoding/decoding algorithm where the code uses this number of bits. (Please give a brief description of how it works on an arbitrary input.)

   The code of the sequence will be $b_1..b_{n-1}$ where $b_i = 0$ if $v_{i+1} = v_i$ and 1 otherwise. To decode, we set $v_1 = 1$, and then set each $v_{i+1} = v_i$ if the i'th code bit is 0 and $v_{i+1} = v_i + 1$ if $b_i = 1$.

   (d) How many slow-growing sequences have last element k? What technique could we use to give an efficient encoding for slow-growing sequences with last element k?

   If the last element is k , there must be exactly $k-1$ times in the sequence where $v_{i+1} = v_i + 1$, i.e., exactly $k-1$ of the $n-1$ bits in the code above are 1. So we could compose the code above with an efficient representation of fixed hamming weight binary strings such as Ranking/Unranking with $n' = n-1$ and $k' = k-1$ as the relevant parameters.

4. Suppose you are an experimental composer and you want to make a melody using 12 notes

$$(A, A\#, B, C, C\#, D, D\#, E, F, F\#, G, G\#)$$

   Each melody is a sequence of 100 notes with the following frequencies:

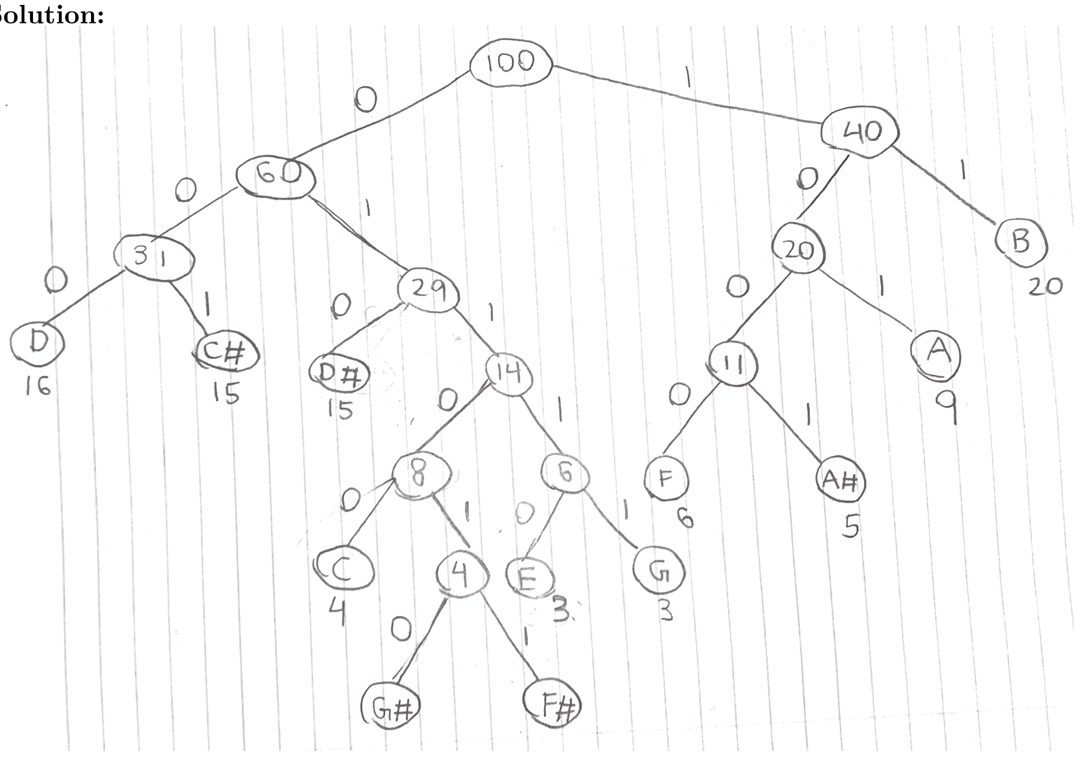
A	A #	B	C	C#	D	D#	E	F	F#	G	G#
9	5	20	4	15	16	15	3	6	1	3	3

(a) Using a *fixed-length* encoding, what is the minimum number of bits needed for each note and how many bits are needed for each melody using this encoding?

There are 12 notes, so to give each a different code requires using $\lceil log_2 12 \rceil = 4$ bits per note. This gives a total of 400 bits in the code.

(b) Draw the huffman tree for the set of frequencies. (you do not have to show intermediate steps.)

Solution:



(c) Give the code for each note. (no justification necessary.)

Solution:

note	frequency	code	length
A	9	101	3
A#	5	1001	4
B	20	11	2
C	4	01100	5
C#	15	001	3
D	16	000	3
D#	15	010	3
E	3	01110	5
F	6	1000	4
F#	1	011011	6
G	3	01111	5
G#	3	011010	6

(d) Calculate the number of bits needed to encode each melody using the huffman encoding. (show how you calculated this.)

Solution:
Multiply the frequency and length of code of each note. Then add them all up:

$$9*3+5*4+20*2+4*5+15*3+16*3+15*3+3*5+6*4+1*6+3*5+3*6 = 323$$

5. Recall that adjacency matrices of *simple* directed graphs are matrices consisting of 0's and 1's such that there are no 1's down the diagonal (no self loops, no parallel edges.)

A directed 3-cycle in a simple directed graph is a set of three distinct vertices x, y, z such that $(x, y) \in E, (y, z) \in E$ and $(z, x) \in E$.

(a) Consider the following algorithm that takes as input an adjacency matrix of a simple directed graph G and returns True if there exists a directed 3-cycle and returns False if there is not a directed 3-cycle.

3-cycle1(G) (G, a simple directed graph with n vertices in adjacency matrix form.)

```
i.   for $i = 1, \dots, n$:
ii.      for $j = 1, \dots, n$:
iii.         if $G[i, j] == 1$ :
iv.              for $k = 1, \dots, n$:
v.                   if $G[j, k] == 1$ and $G[k, i] == 1$:
vi.                      return True
vii. return False
```

(b) (5 points) Show that the runtime for this algorithm is $O(|V|^2 + |V||E|)$.

We run the inner-most for loops once per position where $G[i, j] == 1$, which is at most once per edge (i, j) in the graph. Therefore the total time for the inner-most for loop is $O(|E||V|)$. The rest of the time in the outermost for loops is $O(|V|^2)$, for a total of $O(|V|^2 + |V||E|)$.

(c) (5 points) In order for *3-cycle1* to return True, what needs to happen and why does this correspond to the graph having a directed 3-cycle?

We only return $True$ after first checking that $G[i, j] = 1$ before running the inner-most loop, and then checking that $G[j, k] = 1$ and $G[k, i] = 1$. In this case, $(i, j), (j, k)$ and (i, k) are all in E, so the three edges form a 3-cycle.