Spring 2021 - CMSI 282
Final Exam - ONLINE!

Professor Andrew Forney

BEFORE STARTING ANYTHING:

1. **Place your name in the README.md!**
2. **In the spot for Honor Code Initials in the README.md, place your initials signifying that you agree to the following:**

"I, the above listed, do solemnly swear that I am up to all good, and that I shall not share any exam questions NOR answers on this test with any of my peers, NOR attempt to solicit answers or otherwise copy from anyone else or another source during the examination, lest I be forced to climb a foggy hill made from an objective function for all of eternity."

3. **ALL Answers must be recorded in the Answers.java file in the proper location and format specified – when in doubt, ASK! I will *NOT* grade any answers placed on this PDF!**

Warning: The exam may feel long and you may not finish – remember that there is a difficulty adjustment, so do not panic, and focus on quality over quantity – you're going to be just fine!

Reminder: Exams do not define you or your worth; pass or fail, you'll learn something, or at least learn how much you've learned. This has been a tough year, but you are tougher for sticking with it. No matter what the outcome, you belong here!

Part One – Edit Distance [15p]

Good old Edit Distance – how long has it been? Let's start off with a small variant known as the *Levenshtein Distance*. The Levenshtein Distance is a metric of edit distance that is sometimes preferred outside of spelling correction because it uses only the single character string manipulations; specifically, it has the same recurrence as the definition of edit distance that we saw in class except without the transposition case (meaning we only have insertion, deletion, and substitution as our string manipulations).

1.1. [10p] Complete the memoization table for bottom-up dynamic programming that solves `LevenshteinDistance(R,C)` for $R = "AABCDFE"$ and $C = "ABDCEF"$ as formatted below and in Answer.java.

$\downarrow R, \rightarrow C$	$\emptyset$	**A**	**B**	**D**	**C**	**E**	**F**
$\emptyset$							
A							
A							
B							
C							
D							
F							
E							

1.2. [3p] Suppose we have any arbitrary Strings (R, C) and compute the edit distance using the definition we saw in class that uses all 4 string manipulations of insertion, deletion, substitution and transposition; call this distance E. Were we to compute `LevenshteinDistance(R,C)`, and call this distance L, what would be the relationship between distances E and L that would be true for any arbitrary Strings (R, C)?

A. $E \neq L$ B. $E \leq L$ C. $E \geq L$ D. $E > L$ E. $E < L$

1.3. [2p] True or False: Levenshtein Distance, if computed using top-down dynamic programming has a chance (depending on the input Strings) to leave some cells of the memoization table unsolved. In other words, true or false: there exist some Strings (R, C) for which a top-down deployment of the Levenshtein Distance will leave some cells unsolved?

Part Two – Bloom Filters [15p]

Consider a Bloom Filter with $k = 3$ hash functions f_0, f_1, f_2, a bit array of $m = 16$ bits, and we wish to insert the following items A, B, C:

Item i	$f_0(i)$	$f_1(i)$	$f_2(i)$
A	0	4	12
B	2	4	13
C	9	5	2

2.1. [3p] Show the state of the bloom filter, which we'll call B_0, that results from inserting items A, B, C above:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

2.2. [2p] Now, we are querying B_0 with the following items; for each, determine whether the Bloom Filter's response would be a:

A. True Positive B. True Negative C. False Positive D. False Negative

Problem #	Item i	$f_0(i)$	$f_1(i)$	$f_2(i)$
2.2.1.	C	9	5	2
2.2.2.	D	4	4	13
2.2.3.	E	9	3	13

Suppose we have another Bloom Filter B_1 with the same m, k hash functions as B_0 above.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	0	1	0	0	1	1	1	0	1	1	0	0	1

2.3.[3p] Let a third filter $B_2 = B_0 \cap B_1$ (i.e., B_2 contains all elements that are found in *both* B_0 *and* B_1, the set intersection). Show, based on the above, the configuration of the bits in B_2:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

2.4. [3p] Suppose we have n items in an m-bit Bloom Filter, and it's getting a little packed (i.e., our likelihood of a false positive is growing with each added item). Which of the following is true for reducing this likelihood of a false positive while maintaining the items currently within?

A. Create a new bit array that is $2m$ in size, and for each bit in the original's first m indexes, copy the contents (i.e., bits) of the original bit array into the first $[0, m-1]$ bits of the new, larger bit array.

B. Introduce a new hash function for use with both adding and querying items to the original m-sized bit array.

C. Convert the Bloom Filter into a *counting* Bloom Filter, wherein additions no longer merely flip 0 bits to 1, but instead, increment small integer counters at each index.

D. None of the above

For the next problems, consider that we have declared initially empty Bloom Filters $B_3, B_4, B_5, B_6, B_7, B_8$ all of equal size m and all employing the same set of hash functions, and perform the following operations (Remember: Bloom Filters do not store the actual items of the set; just the bit array representing those hashed items, so any operations like union or intersection operate on those bits). The results of the union and intersection methods produce an equivalently sized Bloom Filter containing the contents of these set operations.

```
B3.add("X"); B3.add("Y"); B3.add("Z");
B4.add("W"); B4.add("X"); B4.add("Y");
B5 = B3.union(B4); // Set Union B3 ∪ B4 using only the bit arrays
B6.add("W"); B6.add("X"); B6.add("Y"); B6.add("Z");
B7 = B3.intersection(B4); // Set Intersection B3 ∩ B4 using only
                          // the bit arrays
B8.add("X"); B8.add("Y");
```

Now, let p_i be the likelihood of a false positive in each of the i-subscripted Bloom Filters above (e.g., p_5 is the likelihood of a false positive in B_5).

2.5. [2p] What is the correct relationship between p_5 and p_6?

A. $p_5 < p_6$ B. $p_5 == p_6$ C. $p_5 > p_6$ D. Can't answer with info given

2.6. [2p] What is the correct relationship between p_7 and p_8?

A. $p_7 <= p_8$ B. $p_7 != p_8$ C. $p_7 >= p_8$ D. Can't answer with info given

Part Three – Compression [15p]

You knew Huffman wasn't going anywhere!

3.1. [10p] Construct an Encoding Map of characters to their compressed bitstring from the following text corpus by constructing the Huffman Trie associated with its character frequencies *using the rules of construction like in Classwork 4; DO NOT include the ETB like in Homework 4.* As a refresher of the rules for how to construct the trie:

- For any ties in frequency, use ascending alphabetic order of earliest letter in its subtree first (i.e., A first, then B, then C, etc.); characters in each leaf count as its "letter in the subtree."
- Nodes popped from the priority queue first produce a 0 bit, and those second, a 1 bit.

Corpus to Compress: `DBCCEEAAAFFF`

Character	Frequency	Bitstring
D	1	
B	1	
C	2	
E	2	
A	3	
F	3	

3.2. [5p] Using the Huffman Trie that you constructed in pursuit of problem 3.1., decode the following bitstring to reveal a word that probably has no meaning to you during finals week. (NB., though the word doesn't have the same character frequencies as the trie constructed above, you can still use it to decode messages with the same characters). Note also: there are no padding 0s like in the Homework. Provide your answer in Answers.java in capital letters.

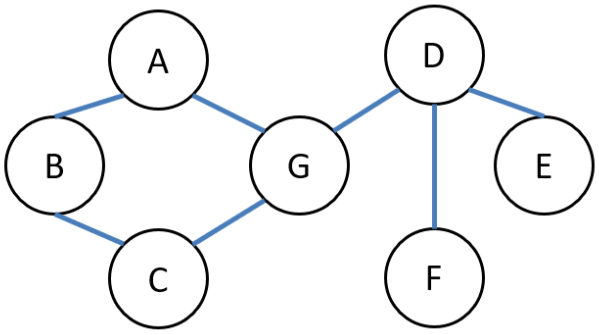
Message to Decode:

`1101001110110`

Part Four – CSP Potpourri [30p]

Suppose we have events that must be scheduled in specific rooms with a day's worth of set-up and cleanup. In this variant of meeting scheduling, we have some events, $\{A, B, C, D, E, F, G\}$ that must be scheduled within a range of days, indexed $[0,4]$, inclusive with the following details:

- Certain events can occur only a restricted set of days (see below for domains D_i).
- Events that are connected in the following constraint graph can neither occur on (a) the same date, *nor* (b) on adjacent dates (± 1). For example, if A, B are adjacent, and $A = 1$, then B cannot occur on days $0, 1$, or 2 because 0 and 2 are consecutively around 1.



Domains					
D_A	0		2	3	4
D_B	0	1	2	3	
D_C	0	1	2		4
D_D	0	1		3	4
D_E	0	1	2	3	4
D_F	0	1		3	4
D_G	0	1		3	4

4.1. [4p] Suppose, during backtracking, we have assigned $\{C = 1, D = 2\}$ and performed *forward checking* to update domains; by the *MRV heuristic*, which variable should we assign-to next? In the event of ties for best, choose the tied variable that occurs earliest in the alphabet.

4.2. [4p] Suppose, during backtracking, we have decided to assign to G first (without any other variables assigned); which of its values should we try first by the *LCV heuristic*? In the event of ties for best, choose the smallest value from amongst those tied.

4.3. [4p] If we are assigning to variables in ascending alphabetic order during backtracking (and attempting to assign values to each variable in ascending order) *with forward checking*, what partial-assignment would come next if we've already assigned $\{A = 0, B = 2\}$?

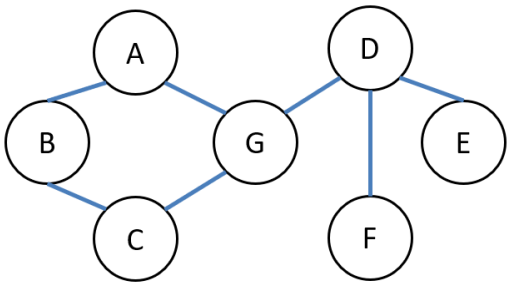
Suppose we are solving the above using Hill Climbing (without simulated annealing), objective function $f(s) = \{\#\text{ constraints satisfied}\}$, and begin with the following random state:

Current State s						
A	**B**	**C**	**D**	**E**	**F**	**G**
0	2	4	1	3	1	1

4.4. [4p] What variables would be candidates for reassignment in the next step of local search?

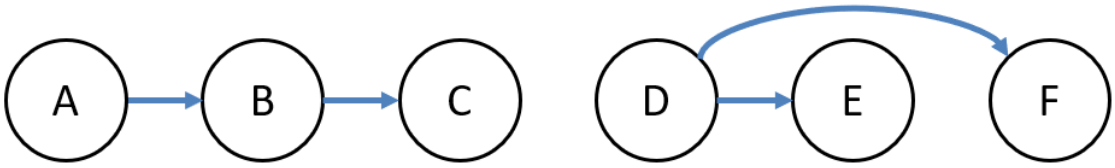
4.5. [4p] If we wished to assign to D in state s by the *Min-Conflict Heuristic*, which value would we assign to it? If there are ties in the best, choose the one with the smallest value.

Noticing from the previous constraint graph that this is a nearly-tree-structured CSP (repeated here for convenience):



Let's try to apply some *directed-arc-consistency* to solve it – the only difference: **we'll have a different set of domains** (displayed in the table below). Additionally, we'll choose $\{G\}$ as our cutset, and assign $G = 0$ as our conditioning in the cutset conditioning algorithm.

4.6. [10p] Find a solution to the CSP by: (1) performing conditioning for $G = 0$ (not yet done for you in the table below), (2) performing directed arc-consistency enforcement, and finally (3) providing an assignment to each of the variables. In the event that multiple variable values will satisfy the constraints during the assignment step, choose the smallest amongst the satisfying options. An outline, including the indexing for each variable, is provided below:

Index	0	1	2	3	4	5
Graph						
Domains	X 1 2 3	X 1 2 X	0 1 2 3	0 X 2 3	X 1 2 3	X 1 X X
Arc Cons.						
Assign						

Part Five – Consistency [15p]

AC-3: Consider the following numerical CSP for the questions following:

- *Variables:* $V = \{W, X, Y, Z\}$
- *Domains:* $D_v = \{0,1,2,3\}\ \forall\ v \in V$
- *Constraints:*
  - $W > Y$
  - $Y = ceil\left(\frac{Z}{2}\right)$
  - $X = Z + 1$
  - $W > X$

(Remember that the ceil function rounds fractions up, and the floor function rounds down).

5.1. [10p] Using the AC-3 Algorithm, show the state of each variables' domain upon the termination of AC-3, leaving only the values remaining in each domain set in Answers.java. In the event that you reduce *any* domain to the empty set (meaning no solution), reduce all domains to the empty set (since none of them will work if any one of them won't).

[!] Warning: this problem may take awhile, so you may want to save it until last.

***k*-Consistency:** For some CSPs, we may not have the luxury of unary and binary constraints, but also ternary, and higher-order ones. We can, in these settings, still check for domain consistency, but must do so between "arcs" in which values in the *tail's domain* must have at least 1 satisfying assignment to *all variables* in the arc's *head*, where the head is now a *set* of variables rather than a single variable. For instance, with the ternary constraint of $X = Y + Z$, we may check that, for a ternary arc $(X \rightarrow \{Y, Z\})$, every value in the domain of *tail* X has at least 1 satisfying assignment from the domains of both Y, Z in the *head* for all possible combinations of Y, Z. This would be an instance of checking 3-consistency (i.e., for $k = 3$).

Consider the CSP specified below that contains a single ternary constraint.

- *Variables:* $V = \{X, Y, Z\}$
- *Domains:* $D_X = \{3,4,6\},\ D_Y = \{2, 3, 6\},\ D_Z = \{1, 3, 4\}$
- *Constraints:* $C = \{X = Y + Z\}$

5.2. [5p] Determine the values that would remain in each domain after enforcing 3-consistency for *only* the ternary arc: $(Y \rightarrow \{X, Z\})$.

Part Six – Genetic Algorithms [10p]

Consider the following numerical CSP (different from the previous problem) for the questions following:

- *Variables:* $V = \{W, X, Y, Z\}$
- *Domains:* $D_v = \{0,1,2,3\}\ \forall\ v \in V$
- *Constraints:* *[!] NOTE: These have changed from the previous problem.*
  - $W < Y$
  - $Y = floor\left(\frac{Z}{2}\right)$
  - $X = Z - 1$
  - $W < X$

6.1. [5p] If we were solving the CSP above using a genetic algorithm with fitness score equal to the number of constraints satisfied per state, determine which of the following states in some example population would have the highest likelihood of selection. In the event of a tie in best, choose the state letter that appears earliest alphabetically.

State	W	X	Y	Z
A	0	2	1	3
B	1	1	3	2
C	2	0	2	1

6.2. [5p] True or False: The following state *could be* a potential member of the immediately next generation from mating pairs of the population in 6.1.

State s'			
W	**X**	**Y**	**Z**
2	1	1	1

Extra Credit!

You can earn a maximum of 3 bonus points if you create an illustration containing a pun or write some amusing word-play (even just a sentence or two) that involves any topic covered on this exam. Attach this in the doc folder of your submission alongside this exam document, in some sort of text, pdf, or image format.