COMPSCI 250 –Introduction to Computation
Second Midterm Spring 2019 – Solutions

Question 1 (20):

a. (10p) Perform the extended Euclidean algorithm on numbers 103 and 40, and find the inverse of 103 modulo 40 and the inverse of 40 modulo 103, if they exist. Clearly state your answer.

We perform the extended Euclidean algorithm: while computing the gcd, we write each successive remainder as linear combination of the given numbers.

$103 = 1 \cdot 103 + 0 \cdot 40$
$40 = 0 \cdot 103 + 1 \cdot 40$
$23 = 1 \cdot 103 + (-2) \cdot 40 \quad (23 = 103 - 2 \cdot 40)$
$17 = (-1) \cdot 103 + 3 \cdot 40 \quad (17 = 40 - 23)$
$6 = 2 \cdot 103 + (-5) \cdot 40 \quad (6 = 23 - 17)$
$5 = -5 \cdot 103 + 13 \cdot 40 \quad (5 = 17 - 2 \cdot 6)$
$1 = 7 \cdot 103 + (-18) \cdot 40 \quad (1 = 6 - 5)$

It is good at this point to check our numbers, indeed $1 = 721 - 720$.

Since gcd(103, 40) = 1, the two inverses exist and are given by the last linear combination: the inverse of 103 modulo 40 is 7, and the inverse of 40 modulo 103 is -18 (or 103 - 18 = 85).

b. (10p) Solve the congruence system $x \equiv 3 \pmod{103}$, $x \equiv 2 \pmod{40}$.

Since we've seen 103 and 40 are relatively prime, we can use the Chinese Remainder Theorem. From the above, we have:

$-18 \cdot 40 \equiv 1 \pmod{103}$, thus $3 \cdot -18 \cdot 40 \equiv 3 \pmod{103}$
$7 \cdot 103 \equiv 1 \pmod{40}$, thus $2 \cdot 7 \cdot 103 \equiv 2 \pmod{40}$.

The solution is $x \equiv 2 \cdot 7 \cdot 103 + 3 \cdot -18 \cdot 40 \pmod{103 \cdot 40}$, thus $x \equiv -718 \pmod{4120}$ or equivalently $x \equiv 3402 \pmod{4120}$. Again, it is good to check our numbers: 3400 is a multiple of 200 (thus of 40), so $3402 \equiv 2 \pmod{40}$. $3402 = 30 * 103 + 3 * 103 + 3 \equiv 3 \pmod{103}$.

Question 2 (20p)

a. (10p) A sequence a_n is defined recursively by $a_1 = 1$, $a_2 = 4$ and $a_n = 2a_{n-1} - a_{n-2} + 2$ for $n > 2$. Prove that $a_n = n^2$ for all positive naturals n.

Proof by strong induction. Base case: $a_1 = 1 = 1^2$, and $a_2 = 4 = 2^2$, so $P(1)$ and $P(2)$ hold. Inductive step: Assume for arbitrary $n \geq 2$, $a_i = i^2$ for all i with $1 \leq i \leq n$. We show that $a_{n+1} = (n+1)^2$.

$a_{n+1} = 2a_n - a_{n-1} + 2 = 2n^2 - (n-1)^2 + 2 = 2n^2 - (n^2 - 2n + 1) + 2 = n^2 + 2n + 1 = (n+1)^2$, as desired. This completes the proof.

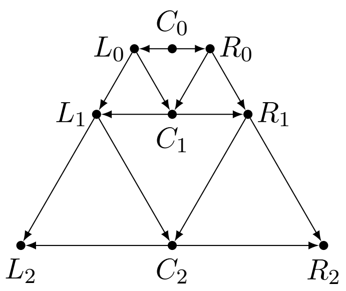
b. (10p) Find a recurrence relation for the number of strings of **0** and **1** with length $n \geq 0$ that do not contain two consecutive **0**s. (Hint: Consider the two choices for the first character).

Let a_n denote the number of bit strings of length n that do not contain two consecutive **0**s. For $n < 2$, all bit strings are valid, thus $a_0 = 1$ (the empty string), $a_1 = 2$. Consider $n \geq 2$.

There are two such types of strings, ending with **1**, or with **0**. The bit strings of length n that end with **1** and do not contain two consecutive **0**s are the bit strings of length $n-1$ with no two consecutive zeros with a **1** added at the end. There are a_{n-1} such strings.

The bit strings of length n that end with **0** and do not have two consecutive **0**s must have **1** as their $(n-1)$st bit, otherwise they have two **0**s at the end. Therefore such bit strings of length n are the bit strings of length $n-2$ with no consecutive **0**s, with **10** added at the end. There are a_{n-2} such strings. Therefore, the total number of strings with no consecutive **0**s is: $a_n = a_{n-1} + a_{n-2}$ for $n \geq 2$, with $a_0 = 1, a_1 = 2$.

Question 3 (50): Consider a directed graph G_n formed of successively larger equilateral triangles, as in the figure (which shows G_2). There are edges from C_k to L_k and R_k (for $k \leq n$), from L_{k-1} to C_k and L_k, and from R_{k-1} to C_k and R_k, if $k > 0$. The length of the segment L_0R_0 is 1.

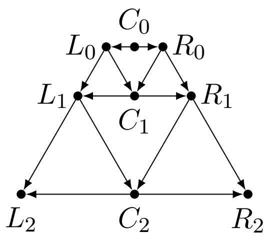


a. (10p) How many paths are there from C_0 to L_n?
Find and justify a recurrence. Then find a formula in terms of n, and prove it by induction.

There is one path from C_0 to L_0. For $n > 0$, L_n has exactly two incoming edges, from L_{n-1} and C_n. Likewise, C_n has two incoming edges, from L_{n-1} and R_{n-1}. Denoting the number of paths from C_0 to these nodes by NC, NL and NR, we get: $NL(n) = NL(n-1) + NC(n)$ and $NC(n) = NL(n-1) + NR(n-1)$, so $NL(n) = 2NL(n-1) + NR(n-1)$. Since the graph is symmetric relative to the center nodes, we clearly have $NR(i) = NL(i)$ for any i.
Our recurrence is therefore: $NL(0) = 1$, $NL(n) = 3NL(n-1)$, for $n > 0$.
This is a geometric series, so we have $NL(n) = 3^n$. We can also easily prove this by induction: for the base case, $NL(0) = 1 = 3^0$ holds. Assuming $NL(n) = 3^n$ for some $n \geq 0$, we have $NL(n+1) = 3NL(n) = 3 \cdot 3^n = 3^{n+1}$, so the inductive goal holds, completing the proof.

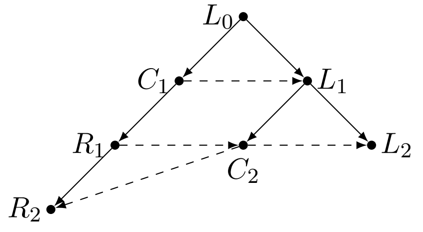
Most saw that there are two paths from L_{n-1} to L_n, either directly or through C_n. Some also saw the paths from the right but found it harder to count them. The most common mistake was to jump and guess a formula only based on the number of paths to $L_0(1)$, $L_1(3)$ and L_2. Many found a wrong count for the paths to L_2 (6, 7, 8) and then tried to "prove" wrong formulas, for instance, $2^n + n$, $2^{n+1} - 1$, or $2^n + n^2$. Since these formulas are wrong, so must be the arguments, though the core of doubling the paths through L_{n-1} is good. Many seemed to try to convince themselves that the formula "makes sense", finding plausible reasons (one more path from the right for $+n$, or for $2(2^n - 1) + 1$, or $2n - 1$ more paths to get to $+n^2$. So directly jumping to conclusions about the formula easily leads to wrong proofs!
The question first asked for a recurrence *and a justification* for a reason. It's not possible to prove a formula by induction just by knowing its value for three points. Even with the right value for L_2 (9), one could still find other "reasonable" formulas that fit, for instance, $2n^2+1$, or $2^{n+2} - 2n - 3$. Even guessing the right formula 3^n (quite easily seen), we have no way to prove the inductive step without reasoning *about the graph.* Some wrote along the lines "assume $NP(n) = 3^n$ paths in G_n; substituting $n+1$ we get $NP(n+1) = 3^{n+1}$ paths in G_{n+1}." This is *not* induction! On the contrary, this is the usual fallacy of proving a statement – here, our $P(n+1)$ – by assuming the same statement! The moment we change n to $n+1$ we go from something known to something we don't yet know. This is also clear in strong induction, where we know everything *up to* a value n and try to prove it for the *next* value $n+1$.

b. (10p) In the graph G_2, carry out a breadth-first search starting from L_0, without goal node, recognizing nodes as they come off the queue. Explore neighbors in alphabetical order. Show the search progress, draw the BFS tree, and identify the non-tree edges.

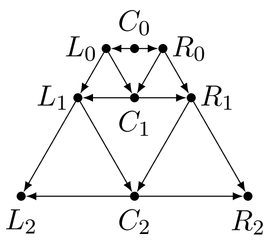


We show the queue contents before adding the neighbors of the node that is taken off.
To draw the edges, we record the predecessor of each node as it is placed on the queue.
Explored nodes are marked canceled as they come off the queue, they don't re-add neighbors.
Edges from their search predecessors are non-tree edges. C_0 and R_0 are not reached.

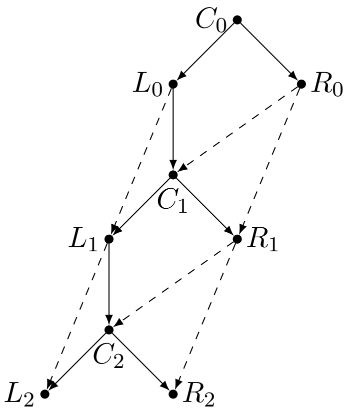
Node	Queue (new nodes bold)
L_0	$\mathbf{C_1(L_0)}, \mathbf{L_1(L_0)}$
$C_1(L_0)$	$L_1(L_0), \mathbf{L_1(C_1)}, \mathbf{R_1(C_1)}$
$L_1(L_0)$	~~$L_1(C_1)$~~, $R_1(C_1), \mathbf{C_2(L_1)}, \mathbf{L_2(L_1)}$
$R_1(C_1)$	$C_2(L_1), L_2(L_1), \mathbf{C_2(R_1)}, \mathbf{R_2(R_1)}$
$C_2(L_1)$	$L_2(L_1), C_2(R_1), R_2(R_1), \mathbf{L_2(C_2)}, \mathbf{R_2(C_2)}$
$L_2(L_1)$	~~$C_2(R_1)$~~, $R_2(R_1), L_2(C_2), R_2(C_2)$
$R_2(R_1)$	~~$L_2(C_2)$~~, ~~$R_2(C_2)$~~



c. (10p) In the graph G_2, carry out a depth-first search starting from C_0, without goal node, recognizing nodes as they come off the stack. Explore neighbors in alphabetical order. Show the search progress, draw the DFS tree, and identify the type of any non-tree edges.



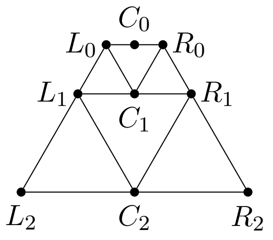
Node	Stack (new nodes bold)
C_0	$\mathbf{L_0(C_0)}, \mathbf{R_0(C_0)}$
$L_0(C_0)$	$\mathbf{C_1(L_0)}, \mathbf{L_1(L_0)}, R_0(C_0)$
$C_1(L_0)$	$\mathbf{L_1(C_1)}, \mathbf{R_1(C_1)}, L_1(L_0), R_0(C_0)$
$L_1(C_1)$	$\mathbf{C_2(L_1)}, \mathbf{L_2(L_1)}, R_1(C_1), L_1(L_0), R_0(C_0)$
$C_2(L_1)$	$\mathbf{L_2(C_2)}, \mathbf{R_2(C_2)}, L_2(L_1), R_1(C_1), L_1(L_0), R_0(C_0)$
$L_2(C_2)$	$R_2(C_2), L_2(L_1), R_1(C_1), L_1(L_0), R_0(C_0)$
$R_2(C_2)$	~~$L_2(L_1)$~~, $R_1(C_1), L_1(L_0), R_0(C_0)$
$R_1(C_1)$	$\mathbf{C_2(R_1)}, \mathbf{R_2(R_1)}, L_1(L_0), R_0(C_0)$
~~$C_2(R_1)$~~	~~$R_2(R_1)$~~, ~~$L_1(L_0)$~~, $R_0(C_0)$
$R_0(C_0)$	$\mathbf{C_1(R_0)}, \mathbf{R_1(R_0)}$
~~$C_1(R_0)$~~	~~$R_1(R_0)$~~



Edges $L_0 \to L_1$ and $L_1 \to L_2$ are forward edges, as they go to descendants.
The other four non-tree edges are cross edges, as they go between different subtrees.
We should check that every edge in the original graph is a tree edge or a non-tree edge.

.

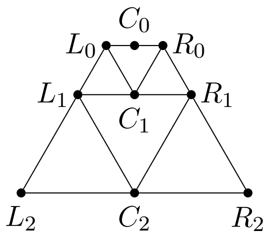
d. (10p) In the **undirected** version of G_2, carry out a uniform-cost search from L_0 with goal node L_2.
Edge costs are equal to their lengths. The length of L_0R_0 is 1.
At equal priority, order nodes alphabetically.



Node	Neighbors	Priority queue (new nodes bold)
$L_0(0)$	$C_0(+.5), C_1(+1), L_1(+1)$	$\mathbf{C_0(.5)}, \mathbf{C_1(1)}, \mathbf{L_1(1)}$
$C_0(.5)$	$\cancel{L_0}, R_0(+.5),$	$C_1(1), L_1(1), \mathbf{R_0(1)}$
$C_1(1)$	$\cancel{L_0}, L_1(+1), R_0(+1), R_1(+1)$	$L_1(1), R_0(1), \mathbf{L_1(2)}, \mathbf{R_0(2)}, \mathbf{R_1(2)}$
$L_1(1)$	$\cancel{L_0}, \cancel{C_1}, C_2(+2), L_2(+2)$	$R_0(1), L_1(2), R_0(2), R_1(2), \mathbf{C_2(3)}, \mathbf{L_2(3)}$
$R_0(1)$	$\cancel{C_0}, \cancel{C_1}, R_1(+1)$	$L_1(2), R_0(2), R_1(2), \mathbf{R_1(2)}, C_2(3), L_2(3)$
		$\cancel{L_1(2)}, \cancel{R_0(2)}, R_1(2), R_1(2), C_2(3), L_2(3)$
$R_1(2)$	$\cancel{C_1}, \cancel{R_0}, C_2(+2), R_2(+2)$	$\cancel{R_1(2)}, C_2(3), L_2(3), \mathbf{C_2(4)}, \mathbf{R_2(4)}$
$C_2(3)$	$\cancel{L_1}, \cancel{R_1}, L_2(+2), R_2(+2)$	$L_2(3), C_2(4), R_2(4), \mathbf{L_2(5)}, \mathbf{R_2(5)}$
$L_2(3)$	goal	$C_2(4), R_2(4), L_2(5), R_2(5)$

We do not re-add nodes that have come off the queue.
We stop when the goal node comes *off* the queue, not when we first reach it.
If we want to recover the path, we must also track predecessors, otherwise we don't need to.
Some chose to draw the search tree; this does not show the order in which nodes were explored.

e. (10p) In the **undirected** version of G_2, carry out an A^* search from L_0 with goal node L_2. The heuristic function is: $h(L_i) = 2 - i$; $h(C_i) = h(L_i) + 1/2$; and $h(R_i) = h(L_i) + 1$.
Edge costs are equal to their lengths. The length of L_0R_0 is 1.
At equal priority, order nodes alphabetically.



Node	Neighbors	Priority queue (new nodes bold)
$L_0(0)$	$L_1(1{+}1), C_1(1{+}1.5), C_0(.5{+}2.5)$	
$L_1(1)$	$L_2(3{+}0), C_1(2{+}1.5), C_2(3{+}.5)$	$C_1(1{+}1.5), C_0(.5{+}2.5)$
$C_1(1)$	$\cancel{L_1}, R_1(2{+}2), R_0(2{+}3)$	$C_0(.5{+}2.5), C_1(2{+}1.5), C_2(3{+}.5)$
$C_0(.5)$	$\cancel{L_0}, R_0(1{+}3)$	$C_1(2{+}1.5), C_2(3{+}.5), R_1(2{+}2), R_0(2{+}3)$
$L_2(3)$	goal	$C_1(2{+}1.5), C_2(3{+}.5), R_0(1{+}3), R_1(2{+}2), R_0(2{+}3)$

The most common mistake was to recognize the goal too early, as soon as it is first seen, and not explore C_1 and C_0 first. Another common mistake is to add the heuristic multiple times. A neighbor R_1 of an explored node $C_1(1 + 1.5)$ is inserted with the cost to C_1 (1) + the cost of C_1R_1 (1) + $h(R_1)(2)$.

Question 4 (20p)

The following are ten true/false questions, with no explanation needed or wanted, no partial credit for wrong answers, and no penalty for guessing.

a. The function $f(m, n) = 2^m(2n + 1)$ is a one-to-one correspondence from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N} \setminus \{0\}$.

TRUE, any positive natural can be written as a power of two times an odd natural (surjection), and by the fundamental theorem of arithmetic, this decomposition is unique (injection).

b. Let P be a property of strings over $\{0, 1\}$. If $P(\lambda)$ is true, and for any string w, $P(w) \to P(w1)$, $P(w) \to P(w00)$ and $P(w0) \to P(w1)$, then $P(w)$ is true for all w.

FALSE, P(0) might not be true

c. Let P be a property of naturals. If $P(0)$ is true, $P(n)$ is true for all odd n, and $P(n) \to P(2n)$ for all n, then $P(n)$ is true for all n.

TRUE, we can get any nonzero even number from an odd number multiplied by a power of two.

d. If the postfix string of an arithmetic expression contains two consecutive operators, then the prefix string of that expression also contains two consecutive operators.

FALSE, consider 1 + 2 * 3. Postfix: 1 2 3 * +. Prefix: + 1 * 2 3.

e. It is possible to construct a game tree with a winning strategy for the first player if less than 1/4 of the leaf nodes are winning nodes.

TRUE, for an arbitrary low fraction. We can have arbitrarily many and large subtrees in which all leaf nodes are losing, and one branch to a subtree with identical winning copies (which we construct in the same way).

f. The height of a rooted tree is the maximum distance from the root to a leaf. Then every rooted binary tree of height h has exactly 2^h leaves.

FALSE, there can be branches of height less than h.

g. Consider the recursive algorithm to cover a 2^k by 2^k square, with one square missing, with L-shaped tiles. The base case of this algorithm is to cover a 1 by 1 square (with one square missing) by doing nothing. When we call this algorithm on a 2^k by 2^k square, the resulting call tree has height k and exactly 4^k leaves.

TRUE, easily seen by induction, which adds one level and 4 subtrees.

h. Since the number 250 satisfies the congruences $x \equiv 16 \pmod{39}$ and $x \equiv 55 \pmod{65}$, the next largest integer to satisfy these two congruences is $250 + 39 \cdot 65 = 2785$.

FALSE, gcd(39,65) = 13. The next largest number is 250 + lcm(39,65) = 250+195=445.

i. If in a connected graph G, after cutting an edge, the resulting graph is no longer connected, then G is a tree.

FALSE, consider a cycle and an extra node connected to it by an edge (which we then cut).

j. A forest is an undirected graph with no cycles. A tree is a connected forest. If F is a forest with five nodes and three edges, there is at most one node in F that is not the endpoint of any edge.

TRUE, there will be trees with 3 + 2 nodes, or 4 + 1 nodes.