NAME: _____

COMPSCI 250
Introduction to Computation
SOLUTIONS to Second Midterm Fall 2021

D. A. M. Barrington and G.Parvini 9 November 2021

DIRECTIONS:

- Answer the problems on the exam pages.
- There are five problems on pages 2-8, some with multiple parts, for 100 total points plus 5 extra credit. Probable scale is somewhere around A=95, C=65, but will be determined after we grade the exam.
- Page 9 contains useful definitions and is given to you separately – do not put answers on it!
- If you need extra space use the back of a page.
- No books, notes, calculators, or collaboration.
- In case of a numerical answer, an arithmetic expression like "$2^{17} - 4$" need not be reduced to a single integer.

Question 1 (10): Recall that the Fibonacci function $F(n)$ is defined by the rules $F(0) = 0$, $F(1) = 1$, and for all n with $n > 1$, $F(n) = F(n-1) + F(n-2)$. Prove that for any natural n, $\sum_{k=1}^{n} F(2k-1) = F(2n)$.

Let $P(n)$ be the statement $\sum_{k=1}^{n} F(2k-1) = F(2n)$. Then $P(0)$ is true because the sum is empty and $F(2 \cdot 0)$ is 0. (We could prove a second case $P(1)$ if we like, showing that the sum is $F(1)$, and noting that $F(1)$ and $F(2)$ are both 1.) For the IH, assume $P(n)$ which says that $\sum_{k=1}^{n} F(2k-1) = F(2n)$. Our IG is $P(n+1)$ which says that $\sum_{k=1}^{n+1} F(2k-1) = F(2n+2)$. The new sum is equal to the first sum plus one more term, which is $F(2(n+1)-1) = F(2n+1)$. This makes the new sum $F(2n) + F(2n+1)$ which is equal to $F(2n+2)$ by the definition of the Fibonacci sequence.

Question 2 (15): Blaze keeps careful observations of how many rabbits she sees on each of her daily evening walks. On Day 0 she saw none, on Day 1 she saw one, and on Day 2 she saw none. She continued her observations for several days, and discovered a pattern in the later numbers. Knowing that recurrence relations have been used to model rabbit populations, she saw that for all n with $n \geq 3$, the number $R(n)$ that she saw on day n was always $R(n) = 3 - R(n-1) - R(n-2) - R(n-3)$.

- (a, 5) Assuming that this pattern continues, compute the values of $R(3)$, $R(4)$, $R(5)$, $R(6)$, and $R(7)$.

By simply evaluating the function each time, $R(3) = 3-0-1-0 = 2$, $R(4) = 3-2-0-1 = 0$, $R(5) = 3-0-2-0 = 1$, $R(6) = 3-1-0-2 = 0$, and $R(7) = 3-0-1-0 = 2$. These answers did not require justification. I took one point off if you correctly solved the wrong problem because of your typo.

- (b, 10) Assuming that this pattern continues, how many rabbits will Blaze observe on Day 503? Prove your answer. *For all naturals n, $R(n) = 0$ if $n \equiv 0 (\text{mod } 4)$, $R(n) = 1$ if $n \equiv 1 (\text{mod } 4)$, $R(n) = 0$ if $n \equiv 2 (\text{mod } 4)$, and $R(n) = 2$ if $n \equiv 3 (\text{mod } 4)$.*

  We proceed by strong induction. For the three base cases $n = 0$, $n = 1$, and $n = 2$ each follow this rule. For larger n, we consider each of the four cases. If $n \equiv 0$, the IH tells us that $R(n-1) = 2$, $R(n-2) = 0$, and $R(n-3) = 1$, so we compute that $R(n) = 3 - 2 - 0 - 1 = 0$, as desired. Similarly, if $n \equiv 1$, we compute $R(n) = 3 - 0 - 2 - 0 = 1$, if $n \equiv 2$, we compute $R(n) = 3 - 1 - 0 - 2 = 0$, and if $n \equiv 3$, we compute $R(n) = 3 - 0 - 1 - 0 = 2$. This completes the strong induction.

  For the specific case of $n = 503$, we can compute that $503 \equiv 3 (\text{mod } 4)$, so $R(503) = 2$ and Blaze would observe two rabbits that day.

  I gave four points out of ten if you didn't even attempt an induction, even if you had $R(503)$ correctly. There were a lot of incorrect induction proofs (usually scored seven out of ten) where you proved the fact you wanted only when $n \equiv 3 \pmod 4$, but used all four cases in your IH.

  There was perhaps a better proof than mine, where you proved $R(n+1) = R(n-3)$ without induction. But to complete that proof, you still need to prove that $R(n) = R(n\%4)$ for all naturals, which needs an induction.

Question 3 (20+5XC): While visiting her childhood home, Alice takes a break from her COMP-SCI 250 homework to sort a box containing some of her n keepsakes. She plans to eventually place each keepsake into a new separate box, but she does this by a series of steps:

- At each step, she takes one non-empty box and divides its items into one non-empty box of p items and one non-empty box of q items.
- During this time, she takes $p \times q$ minutes to reminisce about those keepsakes.
- (For example, dividing a box of seven items into a box of two and a box of five will take $2 \times 5 = 10$ minutes, while dividing into a box of three and a box of four will take $3 \times 4 = 12$ minutes.)
- Later she wonders which choices would have taken the least or most time to get from the single box of n to the final n boxes with one keepsake each.
- She sees that she has only one option for $n \leq 3$, taking 0 time for $n = 1$, 1 minute for $n = 2$, and 3 minutes for $n = 3$.
- For the next two, she sees that however she makes the choices, the total time is 6 minutes for $n = 4$ and 10 minutes for $n = 5$.
- Will this pattern continue? It looks like $T(n) = \frac{n(n-1)}{2)}$.

- (a, 5) Find the possible choices for $n = 6$ and $n = 7$.

For $n = 6$*, we could break into boxes of* 5 *and* 1*, giving us* $5 \cdot 1 + T(5) + T(0) = 10 + 5 + 0 = 10$*, or into boxes of* 4 *and* 2 *for* $4 \cdot 2 + T(4) + T(2) = 8 + 6 + 1 = 15$*, or into boxes of* 3 *and* 3*, for* $3 \cdot 3 + T(3) + T(3) = 9 + 3 + 3 = 15$*.*

For $n = 7$*,* $6 + 1$ *becomes* $6 \cdot 1 + T(6) + T(1) = 6 + 15 + 0 = 21$*,* $5 + 2$ *becomes* $5 \cdot 2 + T(5) + T(2) = 10 + 10 + 1 = 21$*, and* $4 + 3$ *becomes* $4 \cdot 3 + T(4) + T(3) = 12 + 6 + 3 = 21$*.*

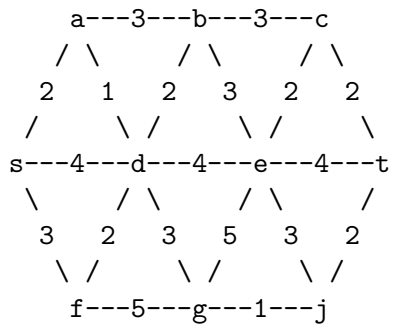
- (b, 15) Prove that for any possible choices of dividing any of the boxes, the total time will be $T(n) = \frac{n(n-1)}{2}$. Use strong induction on all positive naturals, with the base case of $T(1) = 0$.

  As suggested, we use strong induction on n, For $n = 1$, we are told that $T(1) = 0$, and the formula gives us $\frac{1(0)}{2} = 0$. The SIH tells us that for any positive p with $p \leq n$, $T(p) = \frac{p(p-1)}{2}$. To prove out statement $P(n+1)$ from the SIH, we compute $T(n+1)$ by dividing the box with $n+1$ items into boxes of p and q, with $p+q = n+1$, and setting $T(n+1)$ to $pq + \frac{p(p-1)}{2} + \frac{q(q-1)}{2}$. This is $\frac{1}{2}(2pq+p^2-p+q^2-q) = \frac{1}{2}(p^2+2pq+q^2-p-q) = \frac{1}{2}((n+1)^2-(n+1))$. This completes the strong induction.

- (c, 5XC) Suppose that we change the scenario so that Alice spends one minute each time she divides one non-empty box into two non-empty boxes. What is the minimum time for her to get from a single box of n items to n boxes with one item each? Does it matter how she chooses to divide boxes? Prove your answers.

  In this case $T(n) = n-1$. For the base case, we are given $T(1) = 0$. The SIH tells us $T(p) = p-1$ for every positive p with $p \leq n$. To compute $T(n+1)$, with the box of $n+1$ items divided into boxes of p and q, we get $T(n+1) = 1+T(p)+T(q) = 1+p-1+q-1 = p+q-1 = (n+1)-1$.

Question 4 (35): Let G be the weighted undirected graph pictured here:



We want to find the shortest-path distance (using the weights) from s to t. We will also define a heuristic function $h(x)$ for all nodes x in G, such that $h(x)$ will be the smallest number of edges (ignoring the weights) from x to t.

- (a, 5) Compute the value of $h(x)$ for each of the ten nodes x in G, by any method.

  $h(t) = 0$, $h(c) = h(e) = h(j) = 1$, $h(b) = h(d) = h(g) = 2$, and $h(1) = h(s) = h(f) = 3$. We did this by the BFS, starting at t, as described in part (c).

- (b, 5) Which search method, depth-first search or breadth-first search, is useful in solving part (a)? Explain your answer.

  Breadth-first search first finds the neighbors (one edge way), then the their neighbors (two edges away), then three edges away, and so forth. The levels of the BFS tell us how many edges there are from each node to t, and this is defined to be the heuristic value for each node.

- (c, 5) Carry out the search, of the type you chose in (b), for this undirected graph. Describe which nodes enter and leave the open list. Indicate the tree edges and non-tree edges in your search.

  - *t goes on the queue*
  - *t comes off, then c, e, and j go on*
  - *c comes off (from t), b and e go on (the later e will be discarded when it comes off)*
  - *e comes off (from t), b, d, g, j go on*
  - *j comes off (from t), g goes on (note that e is on the closed list)*
  - *b comes off (from c), a and d go on*
  - *the second b is discarded*
  - *d comes off (from e), a, s, and f go on (along with another g to be later discarded)*
  - *g comes off (from e), f goes on*
  - *a comes off (from b), s goes on (to be later discarded)*
  - *s comes off (from d), f goes on (to be later discarded)*
  - *f comes off (from d), nothing more goes on, and (after discarding) we stop*

  The tree edges are indicated above by which node comes off from – they are (t, c), (t, e), (t, j), (c, b), (e, d), (e, g), (b, a), (d, s), and (d, f). Since the graph has 19 edges and a ten-node tree has nine edges, there are ten non-tree edges: (c, e), (e, b), (e, j), (j, g), (b, d), (d, a), (d, g), (g, f), (a, s), and (s, f).

- (d, 10) Trace a uniform-cost search with start node s and goal node t, using the given edge weights, in order to find the shortest-path distance from s to t. Indicate which nodes are on the priority queue at each stage of the search.
  - $(s, 0)$ *goes on*
  - $(s, 0)$ *comes off,* $(a, 2)$*,* $(d, 4)$*, and* $(f, 3)$ *go on*
  - $(a, 2)$ *comes off,* $(b, 5)$*,* $(d, 3)$ *go on*
  - $(f, 3)$ *comes off,* $(d, 5)$*,* $(g, 8)$ *go on*
  - $(d, 3)$ *comes off,* $(b, 5)$*,* $(e, 7)$*,* $(g, 6)$ *go on*
  - *the* $(d, 4)$ *is discarded, we will omit listing the later discards*
  - $(b, 5)$ *comes off,* $(e, 8)$ *and* $(c, 8))$ *go on*
  - $(g, 6))$ *comes off,* $(e, 11)$ *and* $(j, 7)$ *go on*
  - $(e, 7)$ *comes off,* $(c, 9)$*,* $(t, 11)$*,* $(j, 10)$ *go on*
  - $(j, 7)$ *comes off,* $(t, 9)$ *goes on*
  - $(c, 8)$ *comes off,* $(t, 10)$ *goes on*
  - $(t, 9)$ *comes off and we declare victory*

  We find a path of weighted distance 9 *from* s *to* t*. The path is s-a-d-g-j-t. There are a number of arbitrary choices we made here, so there are a number of different valid different runs of the algorithm. There are also a number of incorrect ones.*

- (e, 10) Conduct a complete A^* search of G with start node s and goal node t, using the given edge weights and the given values for the heuristic function h. Indicate which nodes are on the priority queue at each stage of the search.
  - $s0/3$ *goes on (the notation shows its distance of* 0 *from* s*, then the slash after* $h(s)$
  - $s0/3$ *comes off,* $a2/3$*,* $d4/2$*,* $f3/3$ *go on*
  - $a2/3$ *comes off,* $b5/2$*,* $d3/2$ *go on*
  - $d3/2$ *comes off,* $b5/2$*,* $e7/1$*,* $f5/3$*,* $g6/2$ *go on*
  - $f3/3$ *comes off,* $g8/2$ *goes on*
  - $b5/2$ *comes off,* $c8/1$*,* $e8/1$ *go on*
  - $e7/1$ *comes off,* $c9/1$*,* $t11/0$*,* $j10/1$ *go on*
  - $g6/2$ *comes off,* $j7/1$ *goes on*
  - $j7/1$ *comes off,* $t9/0$ *goes on*
  - $t9/0$ *comes off and we declare victory*

  We find the same path s-a-d-g-j-t. It was an arbitrary choice at the last step whether we took $c8/1$ *or* $t9/0$ *first – taking* $t9/0$ *saved us the need to ever look at* c*, but the other choice would look at the same number of nodes. The heuristic isn't very good, which is why we don't save much time in this case.*

Question 5 (20): The following are ten true/false questions, with no explanation needed or wanted, no partial credit for wrong answers, and no penalty for guessing. Some of them refer to the scenarios of the other problems, and/or the entities defined on the supplemental sheet.

- (a) The smallest natural n_0, such that $2^n \geq n^2$ for all $n \geq n_0$, is 0.
  FALSE, $2^n \geq n^2$ is false for $n = 0$ itself. The correct n_0 is 2.

- (b) It follows from the Peano Axioms that if two positive naturals p and q have the same predecessor, then $p = q$.
  TRUE, the Peano Axioms says that a single number cannot have two different successors.

- (c) If we apply a Uniform-Cost Search algorithm on a directed graph where every edge has weight 1, then the resulting search proceeds like a depth-first search.
  FALSE, it acts like breadth-first search, first finding neighbors, then neighbors of neighbors, etc.

- (d) If we carry out a depth-first search from a node s, and the graph contains a directed cycle of more than one node that contains s, then the resulting DFS tree must contain at least one back edge.
  TRUE, the cycle goes from s to itself, and it must start with a tree edges, so it cannot come from an ancestor back to a descendant without using a back edge

- (e) If a state space has finitely many nodes, and no state is ever put on the open list more than once, then a generic search from node s, with t as its goal node, will declare victory if and only if there is a path from s to t.
  TRUE, we know that in a directed search it will be correct if it declares victory or declares defeat, and the additional condition forces it to do one or the other eventually.

- (f) There exist some binary associative operations that are commutative, and other binary associative operations that are not commutative.
  TRUE, addition on naturals meets the first condition, and concatenation on strings meets the other.

- (g) In an undirected tree, the number of nodes is always one less than the number of edges.
  FALSE, the number of nodes is one greater than than the number of edges.

- (h) If w is any binary string, and oc is the one's complement operation, then $oc(0w) = 1oc(w)$.
  TRUE, this was proved on the way to HW#4 – it follows by string induction on w

- (i) In an A^* search, if we use $h(v) = 0$ for every node v in the graph, then the search behaves exactly like a uniform-cost search.
  TRUE, the priority of a node v in A^* is $d(s, v) + h(v)$, and the priority in UCS is just $d(s, v)$.

- (j) If V is the set of nodes in an undirected graph, E is the set of edges, and $deg(v)$ is the degree of a node v, then $\sum_{v \in V} deg(v) = 2|E|$.
  TRUE, when we add the degrees of the nodes, we count every edge exactly twice.