NAME: _____

COMPSCI 250
Introduction to Computation
Second Midterm Spring 2022

D. A. M. Barrington and K.Doney 7 April 2022

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 10 extra credit. Probable scale is somewhere around A=95, C=65, but will be determined after we grade the exam.
- Pages 9 and 10 (printed back to back) 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 $n > 1$, $F(n) = F(n-1) + F(n-2)$. In this problem you are going to solve a *similar*, but *different* recurrence. Here we have two base cases $h(0) = 1$ and $h(1) = 2$, and the general rule for positive naturals n is $h(n+1) = h(n) + 2h(n-1)$. Prove, by strong induction on all naturals n, that $h(n) = 2^n$. You will need two base cases.

Let $P(n)$ be the statement $h(n) = 2^n$. The first base case of $n = 0$ is true because we are given $h(0) = 1$ and $2^0 = 1$. The second base case of $n = 1$ is true because we are given $h(1) = 2$ and $2^1 = 2$. For the general case, assume that $h(i) = 2^i$ for all i with $i \leq n$. The rule tells us that $h(n+1) = h(n) + 2h(n-1)$. Substituting in the SIH, we get $h(n+1) = 2^n + 2(2^{n-1})$, and by arithmetic this is 2^{n+1}. This completes the strong induction.

Question 2 (20+10): Let $\Sigma = \{a, b\}$, so that we are considering strings of a's and b's. A string is defined to have a **triple letter** if it contains either *aaa* or *bbb* as a substring, that is if it ever has the same letter three times in a row. For any natural, let $g(n)$ be the number of strings of length n that *do not* have a triple letter.

- (a, 10) Determine the values of $g(0)$, $g(1)$, $g(2)$, $g(3)$, and $g(4)$ by listing all the elements with no triple letter, of length at most 4. (You might want to compute $g(5)$ but it is not required. Show that $g(3) = g(2) + g(1)$ and $g(4) = g(3) + g(2)$. (You might also want to verify $g(5) = g(4) + g(3)$ but this is not required.)

  $g(0) = 1$ because of the empty string. Since no triple letters can occur with fewer three letters, we have $g(1) = 2$ and $g(2) = 4$. $g(3) = 6$ because the only strings of length 3 with triple letters are aaa and bbb. $g(4) = 10$ because six of the 16 possible strings have a triple letter: aaaa, aaab, abbb, baaa, bbba, and bbbb. The strings of five letters beginning with a are aabaa, aabab, aabba, abaab, ababa, ababb, abbaa, and abbab. There are eight of these, and eight more by swapping a's and b's, giving $g(5) = 16$. We note that $g(3) = g(2) + g(1)$, $g(4) = g(3) + g(2)$, and $g(5) = g(4) + g(3)$.

- (b, 10) Assume that $g(n+1) = g(n) + g(n-1)$ for all n with $n > 1$. Prove, by strong induction on all positive naturals n, that $g(n) = 2F(n+1)$, where F is the ordinary Fibonacci sequence defined in Question 1. You will need two base cases, which you can get from part (a).

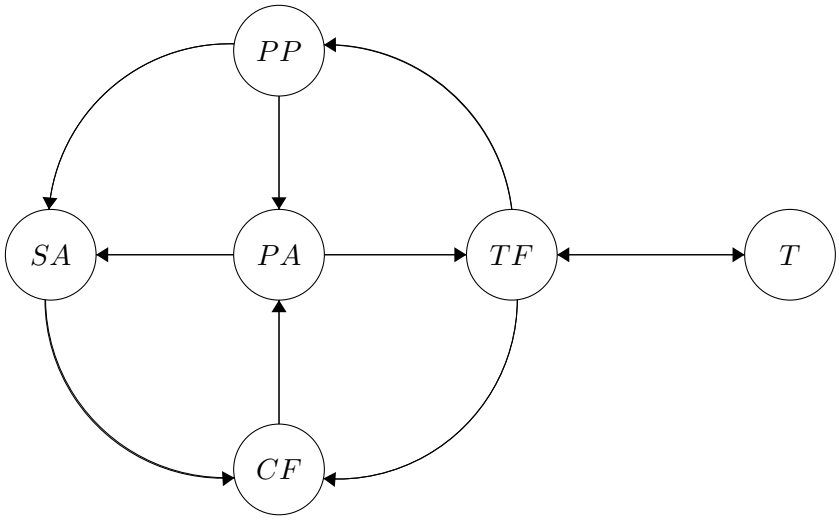
  We have shown $g(1) = 2$ and $g(2) = 4$. and we note that $2F(1+1) = 2$ and $2F(2+1) = 4$. We proceed by strong induction for all positive naturals, with base cases for $n = 1$ and $n = 2$, to prove that $g(n) = 2F(n+1)$. Let n with $n > 1$ and assume as SIH that $g(i) = 2F(i+1)$ for all i such that $1 \leq i \leq n$. We need to prove that $g(n+1) = 2F(n+2$. Our assumption is that $g(n+1) = g(n)+g(n-1)$, which by the SIH is $2F(n+1)+2F(n)$, which is $2F(n+2)$ by the definition of the Fibonacci function.

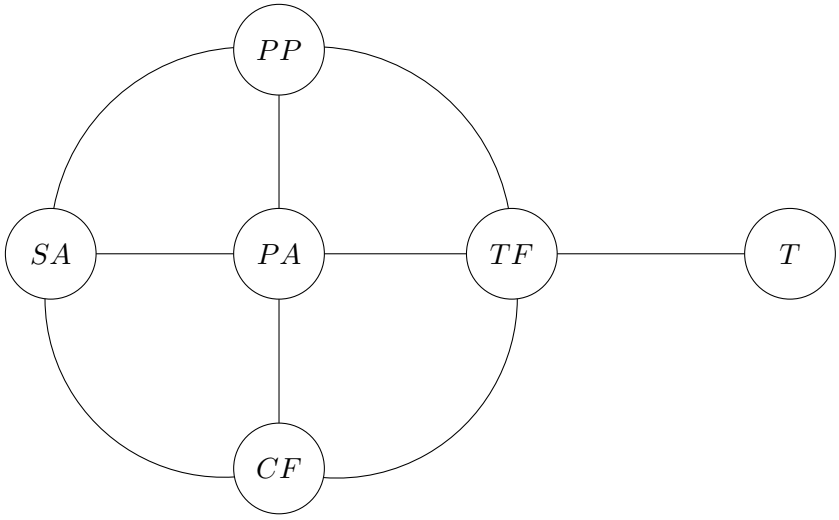
- (c. 10) Prove, for all naturals n with $n > 1$, that $g(n+1) = g(n) + g(n-1)$. (**Hint:** This problem does not necessarily require induction. If you have an arbitrary string of length $n+1$ with no triple letter, look at the case where the last two letters are different and the case where the last two letters are the same.)

  Let $n+1$ be arbitrary with $n > 1$ and consider a string w of length $n+1$ with no triple letter. If w's last two letters are different, let w be written as ua or ub. As we consider all strings w with no triple letter, and last two letters different, these strings have a bijection with strings u with length n and no triple letter. So there are exactly $g(n)$ strings of length $n+1$ with no triple letter and the last two letters differently. Now we consider w's whose last two letters are the same. We can write these strings as vaa or vbb, forming a bijection between these w's and all strings of length $n-1$ with no triple

letter. There are thus exactly $g(n-1)$ of these strings, so that there are $g(n)+g(n-1)$ total strings with length $n+1$ and no triple letter.

Question 3 (20): Pictured here is a directed graph D with six nodes – note that there are edges both from TF to T and from T to TF. Also pictured is the undirected graph U with the same nodes, with undirected edges in place of each directed edge in D.





The two questions follow on the next page.

- (a, 10) Carry out a DFS search for the **directed** graph D starting with node CF. When two or more nodes need to come off the stack and they entered at the same time, take the one first that comes earlier alphabetically. Draw the DFS tree, indicating the non-tree edges, and classify each as a back, cross, or forward edge.
  1. *CF goes on the stack.*
  2. *CF comes off, PA goes on.*
  3. *PA comes off, SA and TF go on.*
  4. *SA comes off, SA-CF becomes a back edge.*

5. *TF comes off, TF-CF becomes a back edge, PP and T go on.*
6. *PP comes off, PP-PA becomes a back edge, PP-SA becomes a cross edge.*
7. *T comes off. T-TF becomes a back edge.*

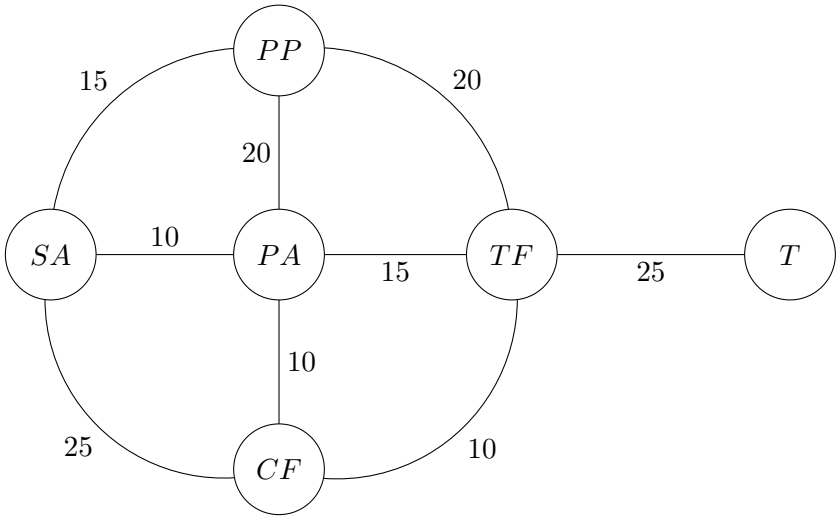
The tree has CF as root, only child PA at level 1, SA and TF are children of PA at level 2, PP and T are children of TF at level 3.

- (b, 10) Carry out a BFS search for the **undirected** graph U, starting with node CF. If two or more nodes need to come off the queue and they entered at the same time, take the one first that comes earlier alphabetically. Draw the BFS tree, indicating the non-tree edges.

1. *CF goes on the queue.*
2. *CF comes off, PA, SA, and TF go on.*
3. *PA comes off, PP, SA, and TF go on.*
4. *SA comes off, SA-PA becomes a non-tree edge, PP goes on.*
5. *TF comes off, TF-PA becomes a non-tree edge, PP goes on , T goes on.*
6. *PP comes off, PP-SA and PP-TF become non-tree edges.*
7. *T comes off, the search terminates*

The BFS tree has CF as the root, PA, SA, and TF on level 1 with tree edges from CF, PP at level 2 with a tree edge from PA, and T at level 2 with a tree edge from TF.

Question 4 (30): Let G be the weighted undirected graph pictured here:



The six nodes in the graph represent locations in Rome, and you are currently at T (the train station) and you want to navigate from there to SA (the Castel Sant'Angelo near your hotel). The edges of the graph indicate walking time in minutes to go from one location to another – for example, it takes 25 minutes to walk from T to TF.

But there is a complication. Your companion insists that whenever we reach one of these locations, you have to stop to visit for the required time in the following table. (So, for example, traveling from T to TF will take you 40 minutes total, 25 for the walk and 15 to visit TF.)

- CF (Campo de Fiori): 25 minutes
- PA (Pantheon): 35 minutes
- PP (Piazza del Popolo): 10 minutes
- SA (Castel Sant'Angelo, your hotel): 10 minutes
- T (Termini, the train station): 0 minutes
- TF (Trevi Fountain): 15 minutes

Your goal is to reach the hotel with the minimum *total* time, and we'll eventually do this with an A^* search using the walking time as a heuristic. We want $h(x)$ for each node to be the walking time from the *goal*, SA, to x.

The two questions follow on the next page.

- (a, 15) Trace a uniform-cost search with start node SA and no goal node, using the walking times given on the graph, to find the value of $h(x)$ for every node x. Indicate which nodes are on the priority queue at each stage of the search.

  1. *(SA, 0) goes on the PQ*
  2. *(SA, 0) comes off, (PA, 10), (PP, 15), and (CF, 25) go on.*
  3. *(PA, 10) comes off, (CF, 20), (TF, 25), and (PP, 30) go on.*
  4. *(PP, 15) comes off, (TF, 35) goes on.*
  5. *(CF, 20) comes off, (TF, 30) goes on.*
  6. *(CF, 25) is discarded.*
  7. *(TF, 25) comes off, (T, 50) goes on.*
  8. *(PP, 30), (TF, 30), and (TF, 35) are all discarded.*
  9. *(T, 50) comes off and the search ends.*

  We have $h(SA) = 0$, $h(PA) = 10$, $h(PP) = 15$, $h(CF) = 20$, $h(TF) = 25$, and $h(T) = 50$.

- (b, 15) Conduct a complete A^* search of G with start node T and goal node SA, using the *total time* for each edge traversed. We determine the added time for each new edge by adding the walking time for that edge and the visit time for the destination. We use the values of heuristic function h from part (a). Indicate which nodes are on the priority queue at each stage of the search.

1. *T(0/50) goes on the PQ.*
2. *T(0/50) comes off, TF(25+15/25) goes on.*
3. *TF(40/25) comes off, CF(50+25/20), PA(55+35/10), and PP(60+10/15) go on.*
4. *PP(70/15) comes off, PA(90+35/10) and SA(95/0) go on.*
5. *There is now a tie in the PQ at 95, if we take CF first, then:*
6. *CF(75/20) goes on, SA(100+10/0) goes on.*
7. *SA(95/0) comes off, and we find the optimal path T-TF-PP-SA*

We have total walking time 25+20+15 = 60 and visit time 15+10+10=35.

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) Let $\phi(x)$ be a predicate on the naturals. If $\phi(0)$ and $\phi(1)$ are true and we have $\forall x : \phi(x) \to (\phi(2x) \vee \phi(2x+1))$ being true, then $\forall x : \phi(x)$ is true.

  FALSE. This is similar to a valid induction, but note the $\vee$ in the conclusion of the implication – if this were $\wedge$, it would work.

- (b) Let $\phi(x)$ be a predicate on the naturals. If $\phi(0)$, $\phi(1)$ and $\phi(2)$ are true and we have $\forall x : \phi(x) \to (\phi(x-1) \to \phi(x+2))$ being true, then $\forall x : \phi(x)$ is true.

  TRUE. We can prove this by strong induction. If we know $\phi(x-1$ and $\phi(x)$, the nested implication proves $\phi(x+1)$, and we have enough base cases.

- (c) Let G be an arbitrary weighted graph with all non-negative weights. Let p_i be the optimal path weight from node i to the goal node g. Let e_i be the minimum number of steps to G from node i to the goal g. The heuristic $h(i) = p_i - e_i$ may fail to be an admissible heuristic for the A* algorithm.

  TRUE. For one thing, p_i might be less than 1, so that $p_i - e_i$ might be negative.

- (d) In the undirected graph U for Question 3, there are exactly two articulation points.

  FALSE. There is only one.

- (e) If m and n are positive integers, then it is not possible to tile every $4m$ by $4n$ rectangle with T shaped tetrominoes.

  FALSE. We can do it by using four of each of them to make 4×4 squares.

- (f) If G is a game with a finite game tree, and every leaf is labeled with -1, 0, or 1, then the value of the game must be -1, 0, or 1.

  TRUE. By induction on all the nodes in the tree, each node must have one of those three values, because each node must be the max or the min of previous values, which each must come from that set.

- (g) Given the arithmetic expression "`+ + * 3 + + 5 7 * 12 6 2 * 7 + 14 1`" in prefix notation, the corresponding infix and postfix notation for the same expression are
  "`3*((5 + 7) + 12 * 6) + 2 + (7 * (14 + 1))`" and
  "`3 5 7 + 12 6 * + * 2 + 7 14 1 + * +`" respectively.

  TRUE. Both claims follow from the recursive translation algorithms.

- (h) To prove a set with two operations form a semiring, it is sufficient to show that both the operators $+, \cdot$ are commutative and associative.

  FALSE. We also need identity elements for both operations, and the distributive law.

- (i) An arbitrary strongly connected directed graph G, with at least two nodes, must have at least one directed cycle.

  TRUE. If x and y are two distinct nodes in G, strong connectivity tells us that there is a path from x to y and a path from y to x. Combining the two gives us a path from x to x, and it has at least two edges in it so it is a directed cycle.

- (j) An undirected graph G is bipartite if and only if there exist no cycles in G.

  FALSE. It is true that if there are no cycles, it is bipartite, but it could be bipartite with cycles if they are all even.