CSE 350 – Theory of Computation (Honors), Spring 2018

Practice Problems

Lecture 1. January 23, 2018

Lecture topic: History of Computation and the Halting Problem.

Problem 1

Write out the halting problem proof on paper for yourself.

No, we cannot write the function *halt(A,I)*.
We can prove this using a proof by contradiction. Let us assume for sake of contradiction that we can write *halt(A,I)* for any program A and input I.
We can then define a function, *uhoh(A)* with the following rule:

uhoh(A):
if halt(A,A) → loop forever
else → halt

This leads to a contradiction where:
if halt(uhoh, uhoh) → uhoh loops
if $\tilde{\text{h}}$alt(uhoh, uhoh) → uhoh halts
Thus we prove we cant write *halt(A,I)* for any A, I.

Problem 2

Consider the halting problem with the constraint that the program and input are run on my Macbook (The Macbook is not connected to the internet.) Can you write $halt(A, I)$ now? Assume no bugs and that you're allowed a computation time longer than the life of the universe.

Yes. Since A,I can be run on a Macbook, it is limited by memory. The states of the memory stored in a computers cache, RAM, and disk are finitely long, so the number of possible unique states are also finitely long. Since the progression of states is deterministic, if a state is repeated, it will enter an infinite loop if the same sequence of states. We know that the machine transitions through more than n states where n is the number of distinct states of the machine, we can determine by the pigeonhole principle that it has repeated at least one state, meaning that it has entered an infinite loop. Therefore we can determine if a program will halt in the time it takes for the Macbook to run through $n + 1$ states.

Lecture 2. January 28, 2018

Lecture topic: Modeling computation: languages.

Problem 3

Determine which of the following are languages and which are not. (Start by reviewing the definition of a language.)

- e - not a language - a language is a set of strings
- $\emptyset$. - language
- $\{\emptyset\}$ - language
- $\{\emptyset, e\}$ - not a language - $\emptyset$ cant be in an alphabet
- The set of natural numbers containing the digit 6. - language
- The set of all valid C programs A and input I. - language
- The set of all valid C programs A and input I where A does not halt on I. - language
- $\{a, b, ab, a\}$. - not a language - 'a' appears twice in the language, going against the definition of a set
- The set of real numbers beginning with the digit 6. - language
- The set of all strings with an equal number of as and bs. - language
- The set of all binary numbers. - language
- Every $s = ab$ where a, b are the concatenation of every element in $c, d \subset \mathbb{N}$, respectively. - language

Problem 4

What are the two techniques that I said would help you through most of the class? Please explain how these techniques work through examples.

- Pigeonhole principle: if there are $n + 1$ pigeons and n holes, at least 1 hole has more than 1 pigeon
  example: A computer with n bits of memory has 2^n possible states. If it undergoes $2^n + 1$ state changes, it is is guaranteed to have visited some state more than once.

- Diagonalization argument: on potentially infinite sets, diagonalization arguments prove the existence of unique element by flipping elements on the diagonal, generating a set unique from all existing sets
  example: write out the set of all binary numbers where each line contains a different number. by taking the digit at the diagonal (i.e. the row of the number and the index of the digit in the number are the same) and flipping it, we create a new binary number where it differs in at least one digit from all existing numbers you have "written out"

Lecture 3. Tuesday, January 30, 2018

Lecture topic: Review of Sets and Relations

Problem 5

Is the relation “is a brother of” symmetric? Is it transitive? Why or why not?

The relation "is a brother of" is not symmetric in the case where there is a female and male sibling. The male is "a brother of" the female, but the female is not a brother of the male. She is his sister.

The relation "is a brother of" is not transitive. When there are two brothers, A and B, A is the brother of B and B is the brother of A. If the relation were transitive, that means A is the brother of A.

Problem 6

Is the relation “is a sibling of” symmetric? Is it transitive? Why or why not?

The relation "is a sibling of" is symmetric. If A is the sibling of B, then B is the sibling of A. It is impossible for one person to be the sibling of another, yet not be regarded with that relation in turn.

The relation "is a sibling of" is not transitive. When there are two siblings, A and B, A is the sibling of B and B is the sibling of A. If the relation were transitive, that means A is the sibling of A.

Problem 7

Is symmetry closed under the union operation? What about transitivity? Prove or show a counterexample.

Symmetry is closed under the union operation. The definition of a union is that if $(a, b) \in R \rightarrow (b, a) \in R$.
$\forall (a, b) \in A \cup B, (a, b) \in A \vee (a, b) \in B$
By the definition of symmetry, $(a, b) \in A, \rightarrow (b, a) \in A \rightarrow (b, a) \in A \cup B$, $(a, b) \in B, \rightarrow (b, a) \in B \rightarrow (b, a) \in A \cup B$
So $\forall (a, b) \in A \cup B, (b, a) \in A \cup B$

Transitivity is not closed under the union operation.
Let $A = \{(a, b), (b, c), (a, c)\}$ and $B = \{(c, d), (d, f), (c, f)\}$. Both sets are transitive. However $A \cup B = \{(a, b), (b, c), (a, c), (c, d), (d, f), (c, f)\}$. This set contains $(b, c), (c, d)$, but does not contain (b, d).

Problem 8 (hard puzzle/challenge problem)

A group of n soldiers is standing in a line. Their sergeant, who stands at one end of the line, wants them all to fire at the same time. Construct a finite automaton to control the soldiers and guarantee that they fire simultaneously.

Note: Having all soldiers fire on the first time step is not a valid solution because there is no first time step. They must begin the firing protocol only after they hear the command.

Hint: What does them having finite memory imply?

O(nlogn) solution:

First, we establish a method that in O(n) time and requires 2 bits of memory per soldier, finds the middle soldier(s) in the line.

The first soldier passes a message that we'll call 'a' down the line. Every soldier who receives it passes it on to the next soldier on the following "turn." When the last soldier receives it, they send it back up the line.

Two turns after the first soldier passes the initial message 'a', they then pass another message 'b' down the line. Every soldier who receives it must wait two turns before passing it on.

The result is that for every three steps 'a' travels down the line, 'b' travels one step. 'b' travels three times slower than 'a', so in the time it takes 'b' to make it halfway down the line, 'a' will have travelled one-and-a-half times the length of the line and be at the halfway point moving back up the line.

If the line has an odd number of soldiers, then the middle soldier will simultaneously receive 'a' and 'b' coming from opposite directions. If the line has an even number of soldiers, then the two middle soldiers may not necessarily receive 'a' and 'b' on the same turn, but if they try and "pass" it along to the next soldier and the soldier is already carrying the other message, then that means the two of them are the middle soldiers.

The time it takes to find the middle soldier(s) is at $\frac{3}{2}n$ or $O(n)$.

Odd-N example below:

a→				
	a→			
b→		a→		
			a→	
				←a
	b→		←a	
		ab		

Even-N example below:

a→					
	a→				
b→		a→			
			a→		
				a→	
	b→				←a
				←a	
		b→	←a		
		b	a		

Now that we can find the middle soldier(s) in $O(n)$ time, we can use this to coordinate the firing.

First, imagine each soldier has a bit that is all initially set to 0. In a method similar to mergesort's or quicksort's halfway partitioning method, we first find the middle soldier(s) of the line and have them set their bit(s) to 1, thus dividing the line into two equal parts on either side of the middle soldier(s).

The soldier(s) in the middle then initiate the same $O(n)$ median-search procedure on each side to find the middle soldier(s) of each half and have them set their bits to 1. Repeat this recursively until all the soldiers have their bits set to 1.

Once a soldier and their 2 neighbors (or 1 neighbor in the case of the ones at the ends) have their bits set to 1, then they fire the following turn. The reason this works is because the procedure always flips the bit of the soldier at the halfway point, guaranteeing that the soldiers end up alternating bits before they all flip to 1.

Example below:

1st Search	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2nd Search	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
3rd Search	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4th Search	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
5th Search	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Surrounding Bits Set - Fire	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The rate at which soldiers are "flipped" doubles every iteration of the search since the middle soldiers each start two new searches. This is a divide-and-conquer strategy that requires $O(log_2 n)$ searches to complete.

Since each search is $O(n)$ and there are $O(log_2 n)$ searches, the total time complexity of this procedure is $O(nlogn)$.

Problem 9 (hard puzzle/challenge problem)
Given a relation R, compute the transitive closure of R in time equivalent to the cost of one matrix multiplication. Assume that the relation R is acyclic.

There exists a method to compute the transitive closure of a relation in $O(logn)$ time where n is the size of the set on R. Firstly, built the nxn adjacency matrix for R where a 1 indicates the presence of the corresponding ordered pair in the set while a 0 indicates a lack thereof.

We then proceed to build a new updated nxn graph, $G_2 = G_1G_1$, where an element at row i and column j, $e_{i,j}$ is equal to row i of G_1 times column j of G_1.

Repeat this where $G_k = G_{k-1}G_{k-1}$ until no new paths are formed (i.e. no changes from 0 to 1). $G_1 \cup G_2 \cup ...G_k$ is the transitive closure, where k is the matrix of all graphs with k-length path mappings.

Lecture 4. Thursday, February 2, 2018

Lecture topic: Review of Relations, functions, and countability.

Problem 10

How do you prove that a set is countable?

We can prove a set is countable by proving there exists an injection or bijection between the elements of that set and the set of natural numbers. For sets of potentially infinite length elements, boustrophedonic traversal methods can be used to index all elements in the set.

Problem 11

Is the union of finitely many countable sets countable? Prove or disprove.

Hint: Use induction.

Yes, the union of finitely many countable sets is countable.

We do this by proving there exists a traversal method that will eventually reach any given element in any of the sets in a finite amount of time. This indexes each element to the natural numbers by the number of steps it takes to reach that element.

We will perform a proof by induction on n.

Let $P(n)$ be the predicate that the union of a finite number n sets is countable.

Base Case: P(1) is true - an element in the union is indexed by its position in the set.

Induction Step: Assume P(n) is true so that the union of n countable sets is countable

We will let S_n be the countable union of n countable sets.

We union one more infinitely countable set S_{n+1} to the S_n.

We can index each element in both S_n and S_{n+1} by alternating between elements in each set. So an element at index j in S_n will be at index $2j$ in the union. An element at index k in S_{n+1} will be at index $2k + 1$ in the union. Since j and k are natural numbers, their index in the union will also be natural numbers. Any given element in either set will be matched to the natural number and eventually reached.

Therefore $P(n + 1)$ is true - the union of $n + 1$ countable sets is also countable.

Since the base case and the induction step are true, $P(n)$

Problem 12

The inductive strategy works for the union of finitely many countable sets, but breaks down on the union of a countably infinite number of countable sets. Why?

Hint: Think about what the predicate to be used in the induction was.

The inductive strategy breaks down for the union of a countably infinite number of countable sets because we assume P(n) is true. However, n must be a finite number, otherwise we can't assume that the predicate is true for a number n that is infinity.