CS 332
Spring 2018

CS 332 - Homework 4, with brief answers

Due: Thursday, March 29

Reading: Chapter 5.1 and 5.3 in the Sipser textbook, pages 213 - 216 and 234-238. So skip section 5.2 and also those results in 5.1 which refer back to Section 1 of the book. Chapter 7, pages 275 - 298.

PROBLEMS
Part 1: These 6 problems are to be turned in and each will be graded (10 points each).

1. Recall Dijkstra’s algorithm. This algorithm takes as input an undirected, weighted graph G with N vertices and a specific vertex s of G. It outputs the length of the shortest path from s to any other vertex of G.

   If you never learned this algorithm you should have. Look it up in your favorite algorithm book and see how it works. It’s pretty easy to understand, but go over it and make sure you see how it works.

   i. What is the complexity (of big-O notation) of Dijkstra’s algorithm? You need not justify your answer but you should write where you found your version of the algorithm.

   Answer: The usual versions of Dijkstra’s algorithm run it time $O(n^2)$.

   ii. Now consider the “all pairs shortest path” problem. This problem takes the same kind of input graph G as Dijkstra, but it tell you more. Namely, for each pair of vertices u, v from G it outputs the length of the shortest path from u to v.

   Show how you can reduce the “all pairs shortest path” problem to Dijkstra’s algorithm.

   That is, give an algorithm which solves “all pairs shortest path” and which uses Dijkstra’s algorithm as a procedure call (a subroutine).

   What is the complexity of your algorithm?

   Answer: To solve the all pairs problem you just need to run Dijkstra n times, each time starting at a different vertex in G. This will result in the length of the shortest path from u to v for all u,v in G. The running time will just be $n \times O(n^2)$ which is $O(n^3)$.

   iii. Give a lower bound for the “all pairs shortest path” problem

   That is, find a function f(n) for which it would not be possible to solve the problem in less than $O(f(n))$. (I’m not looking for any difficult new algorithm here, just for a “trivial” f which any algorithm A which solves the problem cannot be smaller than.) Your f should be strictly smaller than the complexity of the algorithm you found in part ii.

   Answer:

   $f(n) = n^2$ is a lower bound, since any answer to the all pairs problem consists $O(n^2)$ outputs, one for each pair u,v of vertices in G. Since each path length takes at least one step of the algorithm to output you cannot have a solution to the all pairs shortest path problem in less than $O(n^2)$ steps.
2. Problem 7.8 on page 323 of the Sipser textbook.

Answer:

Take a look at the graph connectivity algorithm on page 186. You need to go through the 4 steps and estimate for each step how much time (that is how many steps) this will take to carry out on a graph with \( n \) nodes.

Step 1: Select \( v_1 \) and mark it and initialize a boolean list \( n \) bits to all 0’s, and set a flag variable \( N \) to true. Total of \( 3+n \) steps.

(\( N \) being true indicates that there is a newly marked node. As we go through the algorithm we will set vertices of \( G \) to 1 in the boolean list when they get marked.)

Step 2: To do the test for “some newly marked nodes” test \( N \) to see if it is true. If \( N \) is true then carry out step 3, else go to step 4. (Total of 2 steps each time it is executed.)

(This step is carried out at most \( n \) times.)

Step 3: (This step is carried out at most \( n \) times.)

Set \( N=\text{false}\). Go through the list of \( G \)’s nodes, and for each unmarked node \( v \) look at \( v \)’s edges and see if \( v \) is attached to some marked node. If so you mark \( v \) by setting the bit corresponding to \( v \) in the Boolean list to 1. When this occurs the variable \( N \) is is set to true.

Total of \( 1 + O(n \times (n+1)) \) steps.

Step 4: It takes \( n \) steps to see if any node in \( G \) is unmarked and one more step to accept or reject.

Total steps for the whole algorithm (worst case): \((3+n) + (2n) + (1+n \times (n+1)) + (n+1)\) which is \( O(n^2) \)

3. Say whether each of i. - iv. is True or False. In each case briefly justify your answer.

i. There are infinitely many undecidable problems.

Answer:

i. True. One way to see this is to take one element at a time out of an infinite undecidable set (like \( HALT_{TM} \)). Each of the resulting sets is still undecidable and there are infinitely many of them.

ii. If \( L \) is decidable and \( J \) is undecidable then \( L \cup J \) is undecidable.

Answer:

ii. False. Let \( L = \Sigma^* \) and \( J = HALT_{TM} \). \( L \) is decidable and \( J \) is undecidable but \( L \cup J = \Sigma^* = L \) which is decidable.

iii. If \( L \) is decidable and \( J \) is undecidable then \( L \cap J \) is undecidable.

Answer:

iii. False. Let \( L = \emptyset \) and \( J = HALT_{TM} \). \( L \) is decidable and \( J \) is undecidable but \( L \cap J = \emptyset = L \) which is decidable.

iv. If \( J \) is reducible to \( K \) and \( K \) is recognizable then \( J \) is also recognizable.

Answer:

iv. True. Construct the recognizer for \( J \) by reducing \( J \) to \( K \) and then running \( K \)’s recognizer.

Note: This proof uses the fact that \( w \in J \) if and only if \( f(w) \in K \). You should state this and also why it is sufficient.
4. Prove that the problem of deciding if a TM M ever writes the symbol 1 on its tape is undecidable.

You should do these problems using reductions.

Answer:
Let \( L = \{\langle M \rangle \mid M \text{ writes the symbol 1 on its tape} \} \). Recall that if \( X \leq L \) and \( X \) is undecidable, then \( L \) is also undecidable. So, to show that \( L \) is undecidable we will reduce an undecidable problem to \( L \).

Let’s reduce \( A_{TM} \) to \( L \). Suppose \( L \) is decidable and \( M_L \) decides \( L \). We can build \( M_A \) which decides \( A_{TM} \) as follows:

\[ M_A = \text{"On input } \langle M, w \rangle \text{:"} \]
1. Construct the TM \( M' \) where \( M' \) ignores its input and runs \( M \) on \( w \) with the modification that where \( M \) uses the symbol 1, \( M' \) uses \( \overline{1} \) instead. \( M' \) also differs from \( M \) in that immediately before entering the accept state \( M' \) writes a 1.
2. Run \( M_L \) on \( M' \). If \( M_L \) accepts, accept. Otherwise, reject.

If \( M_L \) accepts \( M' \), \( M' \) writes a 1 on its tape so \( M \) must accept \( w \) since \( M' \) only writes a 1 immediately before accepting. If \( M_L \) doesn’t accept \( M' \), \( M' \) doesn’t write a 1 so \( M \) must not accept \( w \).

\( M_A \) deciding \( A_{TM} \) is a contradiction since \( A_{TM} \) is undecidable. Therefore, \( L \) is undecidable.

5. Prove that it is undecidable to determine if two TM’s \( M \) and \( L \) accept the same languages.

(Note: So the language which you are to prove undecidable is
\[ S = \{ < M, L > \mid \text{M and L accept exactly the same set of strings } \} . \]

You should do this problem using reductions.

Answer:
Let’s reduce \( A_{TM} \) to \( S \). Suppose \( S \) is decidable and \( M_S \) decides \( S \). We can build \( M_A \) which decides \( A_{TM} \) as follows:

\[ M_A = \text{"On input } \langle M, w \rangle \text{:"} \]
1. Construct machine \( X \) which ignores its input and accepts.
2. Construct machine \( Y \) which ignores its input and runs \( M \) on \( w \).
3. Run \( M_S \) on \( \langle X, Y \rangle \). If \( M_S \) accepts, accept. Otherwise, reject.

If \( M_S \) accepts \( \langle X, Y \rangle \), \( X \) and \( Y \) accept the same language and thus \( Y \) is also accepting every input. This means that \( M \) accepts \( w \) and \( M_A \) should accept. If \( M_S \) rejects, \( M \) must not accept \( w \) and \( M_A \) should reject.

\( M_A \) deciding \( A_{TM} \) is a contradiction since \( A_{TM} \) is undecidable. Therefore, \( S \) is undecidable.

6. You given two languages \( J \) and \( L \) both of which are sets of natural numbers in \( P \).

i. Show that \( J-L \) is in \( P \)

ii. Show that \( JL \) (the concatenation of \( J \) and \( L \)) is in \( P \).

Answer:
\( J \) and \( L \) are both in \( P \) so we can assume that there is a TM \( M_J \) which decides \( J \) in polynomial time \( O(n^c) \) and \( M_L \) which decides \( L \) in polynomial time \( O(n^d) \).
i. 
\( M_{J-L} = \) “On input \( w \):
1. Run \( M_J \) on \( w \). If \( M_J \) rejects, reject.
2. Run \( M_L \) on \( w \). If \( M_L \) rejects, accept. Otherwise, reject.”

\( M_{J-L} \) decides \( J - L \) and has running time \( O(n^c + n^d) \) which is polynomial, so \( J - L \) is also in \( P \).

ii. 
\( M_{J_L} = \) “On input \( w = w_1w_2w_3...w_n \):
1. for \( i = 1, \ldots, n - 1 \): 
2. Run \( M_J \) on \( w_1...w_i \). If \( M_J \) rejects, go to 4.
3. Run \( M_L \) on \( w_{i+1}...w_n \). If \( M_L \) accepts, accept.
4. \( i = i + 1 \)
5. Run \( M_J \) on the empty string and \( M_L \) on \( w \). If both accept, accept.
6. Run \( M_J \) on \( w \) and \( M_L \) on the empty string. If both accept, accept.
7. Reject.”

\( M_{J_L} \) decides \( J_L \) and has running time \( O(n(n^c + n^d)) \) which is polynomial, so \( J_L \) is also in \( P \).

Part 2: These problems are good practice and you should try them. They will not be graded

1. Note: Some of the ideas for this problem were covered in section. You can see the exact definitions you need on page 271 of our textbook.
   a. Give an example of a Boolean formula \( F \) which contains at least 2 different variables, and where \( F \) is satisfiable and its negation \( \neg F \) is not satisfiable.
   b. Give an example of a Boolean formula \( F \) which contains at least 3 different variables, and where \( F \) is satisfiable and has exactly 4 satisfying truth assignments.

2. Page 239, problem 5.7

3. Page 239, problem 5.8