CSC 379 Homework 5

UM-Flint, Winter 2019

Some problems in this assignment taken from Introduction to Algorithms 3rd Edition, Cormen et. Al.

Due Date: Thursday, 3/28/20198, 11:59 PM EST. Late assignments not accepted without an approved excuse.

Submission Format: Submit typed as a pdf file. Use the Attach File option through Blackboard, on this assignment.

I will not accept hand-written assignments. PDF submissions **may not include images of typed text**. Any text which is contained within an image file **will not be graded**. Yes, I can tell the difference. You may use image files for any diagrams you need to draw, however.

Scoring:

Maximum Points Possible: 100 (+5 extra credit)

Group Member Rules: You may have **unlimited** partners at no cost**.**

Each person must submit his or her own copy of the assignment to receive credit. You are not required to have a partner. You can freely chat on the discord server about questions and solutions. If someone (other than the instructor) is particularly helpful to you, please cite them on your assignment.

However, and this is important: **Please do not simply give solutions away.** The reason is that if someone does not understand the solutions, does not practice on their own, etc. this means they will A) not develop a mastery of the material and B) **probably not do well on the exams.**

For problems where a solution is provided in advance, you should complete the problem as far as you normally would have before turning it in. Then, look up the solution provided and write up your analysis. You can still get full credit for an incorrect original solution if you provide good analysis. You may lose points if your original work is incomplete, not provided at all, or does not show sufficient progress before looking things up. I will have to arbitrate on what is "sufficient progress," but again you should wait to look things up after you are satisfied with what you have done.

This is an experiment. I want to see how it goes letting you work together more freely. You are adults and I want to try to treat you as such. So, to that end, on this assignment I will trust that you will endeavor to understand things rather than just get quick solutions without understanding them etc. If the exams show a lower level of understanding on this part of the material I will probably go back to fewer partners. Impress me with how you handle the freedom!

Problem 1) (20 points)

Define each of the following (2 points each):

Decision Problem

Decision problems are problems that with any given input the output is yes or no depending on decision wanting to be made. It is also a deterministic problem.

Verification Problem

Verification problem is still a yes or answer however, its input is a guess and outputs yes or no depending on if guess inputted is a solution to the problem. A verification problem must be checked in polynomial time. It is a non-deterministic problem.

Optimization Problem

Optimization problems are not a yes or no problem, but it is a problem that asks for the solution that gives the best value given a certain criteria.

Polynomial time

Polynomial time is anything that can be done in (n^c) were n is the number of inputs

Pseudo-polynomial time

Pseudo-polynomial time dependent on both the number of inputs and the magnitude of those inputs.

The class P

Is a subclass of the decision problem, but the yes or no answer question can be solved in polynomial time.

The class NP

Is a subclass of the class p. The class NP is a decision problem that is polynomial time that is a verification problem. Were the input is a guess and the goal is to verify if it is a solution or not. NP means nondeterministic polynomial time.

NP-hard

NP-hard is a np problem were all other np problems reduce to it. It can be found by reducing from a NP-Complete problem.

NP-Complete

NP-Complete problems are the hardest problems in the language were all NP problems can be reduced to these NP-Complete problems. In order for something to be NP-Complete it must be NP along with also being NP-Hard

SUBSET SUM (the decision problem)

The subset sum problem is a NP-Complete problem that asks if there is a subset of S that adds up to T. It is a decision problem so it is a yes or no output.

Problem 2) (5 points)

Are the $O(nW)$ dynamic programming algorithms for Gold Split and 0-1 Knapsack polynomial time algorithms? Explain why or why not.

Gold Spilt and 0-1 Knapsack are not polynomial time algorithms they are pseudopolynomial time algorithm because not only do they depend on the size of the input, but they also depend on the magnitude of the input it self. This is because of the size of pack for both the gold split and 0-1 knapsack determine the amount of loops that the algorithms has to go through while also going through loops for each and every time that is given for the input. This makes it O(nW) were n is the amount of items given and W is the pack size.

Problem 3) (25 points) [Textbook Problem: page 1061, 34.1-5]

Show that if an algorithm F makes at most a constant number of calls to polynomial time subroutines and performs an additional amount of work that also takes polynomial time, then F runs in polynomial time [do this part with a proof by induction].

Original Solution

Well, if the subroutine is polynomial it does not matter it will still be polynomial time no matter how many times it calls the subroutine. The only thing that changes to the running time is the coefficient in front of the time complexity of the subroutine function, so basically k multiplied by the time complexity of the subroutine. So, the subroutine is a polynomial no matter what k is, because a polynomial times a constant is still a polynomial. Next the after the subroutine it does a more work that is polynomial time, this is not nested into the subroutine so the extra work that the function does is not multiplicative to the subroutine function it is additive instead. The time complexity would be for the algorithm F would be the polynomial of the subroutine multiplied by k plus the polynomial of the extra work. A polynomial plus a polynomial is still a polynomial.

Modified

Subroutine polynomial multiplied by k if k is equal 0 then the run time is polynomial time from the second portion of the algorithm. However, if k = n and n multiplied by the subroutine polynomial time is an polynomial, then n+1 multiplied by the subroutines polynomial time is still a polynomial. Therefore for any constant k the subroutine run time will be polynomial. The second portion of the algorithm that does another polynomial time complexity work it will just add to the subroutine polynomial. A polynomial plus a polynomial is still a polynomial. Therefore the run time for algorithm F is polynomial time.

Also show that a polynomial number of calls to polynomial-time subroutines may result in an exponential-time algorithm [do this part by showing a single example].

An example is x * x * x..... x times so it will be x * x * x *x *x .... = x^x, this will no longer be in polynomial time. This will be in exponential time.

Explanation

While proving that the algorithm was polynomial I did not use induction to show that a polynomial times a constant was a polynomial, however, I wrote another proof that incorporated induction in the solving the proof for the algorithm. For the showing that a polynomial multiplied by a polynomial could equal a exponential time complexity I think that any polynomial multiplied by x times would result in a exponential. I am a bit confused by the solution used such a complex way to show that it could be exponential. I learned how better do induction along with hopefully better understand how polynomial works and how using the different operations could keep the polynomial a polynomial or change it drastically.

For this problem, there is a solution available online. I wish for you to first attempt to solve the problem yourself. Then, check your work.

Show your original solution from before you looked at the online solution. Then explain how you needed to correct your original solution, if at all. Explain what you learned and how your process has been improved.

When you are ready to compare your work a solution can be found here [Exercise 34-1.5]:

http://sites.math.rutgers.edu/~ajl213/CLRS/Ch34.pdf

Problem 4) (25 points)

A *Hamiltonian cycle* in a graph is a simple circuit that visits every vertex exactly once and then returns to the original vertex.

The HAM-CYCLE problem is: GIVEN a graph G = <V,E>, IS THERE a Hamiltonian cycle in G?

Show the HAM-CYCLE is in NP.

In order to show that HAM-CYCLE is NP it must follow a few conditions. It must have an input of a possible solution, it must output a yes or no, it must also ask if the input is a solution and it must be polynomial time. The first two conditions are meet with the prompt, so we need to show that HAM-CYCLE can be verified in polynomial time. In order to that we need to go through the solution vertex by vertex and make sure that there is an edge that connects both vertex, along with making sure that there is no vertex that have been visited twice except the first and last node in the Hamiltonian cycle. Additionally, we have to make sure that every node has been visited. This can be simply done by looking at all the vertex which can be done in polynomial time.

Explanation

When looking at the answer I was not clear on my description of how to verify the solution that it goes through every vertex in the graph no the solution. However, besides that my solution is similar to the book's proof. I learned that I need to be more descriptive when describing the verification process.

For this problem, there is a solution available in the textbook. I wish for you to first attempt to solve the problem yourself. Then, check your work.

Show your original solution from before you looked at the online solution. Then explain how you needed to correct your original solution, if at all. Explain what you learned and how your process has been improved.

When you are ready to compare your work a solution can be found on page 1091 of the textbook as the first part of the proof of NP-completeness.

Problem 5) (25 points) [Textbook Problem: page 1077, 34.3-2]

Show that the $\leq_P$ relation is a transitive relation on languages (problem classes). That is, show that if $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$, then $L_1 \leq_P L_3$.

Original

To show that $\leq_P$ is transitive you can look at $L_1 \leq_P L_2$ were L_1 were it is upper bounded by L_2. Therefore the upper bounds of L_1 is the upper bounds of L_2. So we can add more information of we knew the upper bound of L_2. Since $L_2 \leq_P L_3$ that means that L_3 is the upper bound of L_2. That means that the upper bound of L_1 is L_3 because the upper bound of L_1 is L_2 and the upper bound of L_2 is L_3. Furthermore, because L_1 can be less or equal to L_2 and L_2 can be less than or equal to L_3 therefore L_1 can be less than or equal to L_3.

Edited

Because L_1 polynomial time reduction of L_2 means that L_1 is not as hard or is hard as L_2. If L_2 is polynomial time reduction of L_3 then L_2 is not as hard or is hard as L_3. Further L_1 is upper bounded by L_2 and that is upper bounded by L_3 means that L_1 is upper bounded by L_3.

Explanation

Well, I looked at the book and I was a bit confused. So, I did not do anything similar to the books solution. However, I think my explanation is pretty good, however, I will change it a bit to make it more understandable.

For this problem, there is a solution available online. I wish for you to first attempt to solve the problem yourself. Then, check your work.

Show your original solution from before you looked at the online solution. Then explain how you needed to correct your original solution, if at all. Explain what you learned and how your process has been improved.

When you are ready to compare your work a solution can be found here [Exercise 34.3-2]:

http://sites.math.rutgers.edu/~ajl213/CLRS/Ch34.pdf

Extra Credit) (+5 points) [Textbook Problem: page 1101, 34.5-6]

A *Hamiltonian path* in a graph is a simple path that visits every vertex from u to v exactly once.

The HAM-PATH problem is: GIVEN a graph G = <V,E> and a pair of vertices $u,v \in V$, IS THERE a Hamiltonian path from u to v?

Show that HAM-PATH is NP-complete. For the NP-hardness part of the proof, use a reduction from HAM-CYCLE.

In order to show that HAM-PATH is NP-complete, we must show that it is NP and NP-hard. Given the prompt it is an NP problem. We just need to make sure that it is a NP-hard. In order to that I will reduce from HAM-CYCLE. If HAM-CYCLE reduces to HAM-PATH then there is some function that will do HAM-PATH type problems. Next we can show that we can solve HAM-PATH using HAM-CYCLE. We can do this by cutting up the HAM-CYCLE in to smaller chunks and use the begging and ending vertex of those chunks as the input for HAM-PATH. Afterward putting them all together will create the output of HAM-CYCLE. This modification can be done in polynomial time because not only do we not increase the amount of vertex that we are checking, but we keep the polynomial time about same. This shows that HAM-CYCLE $\leq_P$ HAM-PATH which means that HAM-PATH is NP-hard and that means that HAM-PATH is NP-complete.

Explanation

I think that my reasoning is similar to what the solution says, however, the solution gives a lot more indepth mathematic solution to the problem, while mine is a bit more of kinda psuedocodeish. However, I learned that I need to look further in to how to improve my polynomial time reductions.

For this problem, there is a solution available online. I wish for you to first attempt to solve the problem yourself. Then, check your work.

Show your original solution from before you looked at the online solution. Then explain how you needed to correct your original solution, if at all. Explain what you learned and how your process has been improved.

When you are ready to compare your work a solution can be found here [Exercise 34.5-6]:

http://sites.math.rutgers.edu/~ajl213/CLRS/Ch34.pdf

Note: The solution here used the term “certificate” to describe the guess of a solution for the proof that HAM-PATH is in NP.