CS 70 Discrete Mathematics and Probability Theory
Summer 2020 Course Notes

Homework 2

Due: Sunday, July 5, 10:00 pm
Grace period until Sunday, July 5, 11:59 pm

1 Exam Policy and Practice

Please read the exam policy carefully before proceeding. This question is designed to familiarize you with some of the things you will have to do during the exam.

1. Fill out the following Google Form to submit the Zoom link you will be using:

   You must use this Zoom link for this assignment, as well as for the exams.

2. Start the Zoom call whose link you provided above. Turn on your microphone and webcam. Turn off your speaker. Share your entire desktop (not just a particular window).

3. Start recording via Zoom. Record locally so that in the event of your internet connection dying your meeting still continues to be recorded.

4. Hold your CalID next to your face and record yourself saying your name into the webcam. Both your face and your entire CalID should be visible in the video. We should be able to read your name and SID. This step should take **at least** 3 seconds. See figure 1. *If you do not have a CalID for some reason, please hold up some document which has an image of you and proves your identity, such as a driver's license.*

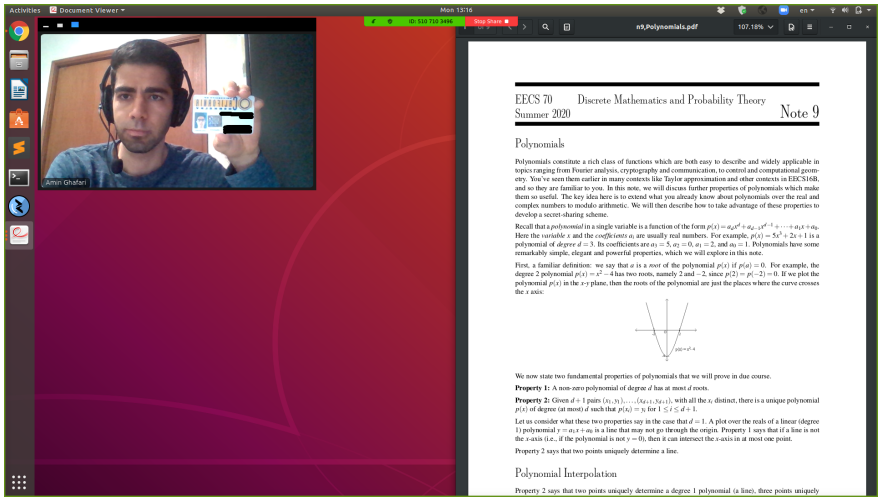


Figure 1: ID card demonstration. Do not actually black out your SID and name.

5. Turn your webcam/laptop around 360° **slowly** so that we can see your entire room clearly. There should be no uncovered screens anywhere in the room during your exam. Only admin TAs and instructors will be able to see your videos (both for this assignment and for the actual exams).

6. Position your webcam/laptop in such a way that we can see your upper body from your head to your hands. Your face and hands must be in the frame at all times. Your phone should also be in the frame at all times, turned upside down. If you do not have sufficient space to include your head, then please make sure to include your hands, phone, and as much of your torso as possible. See figure 2.

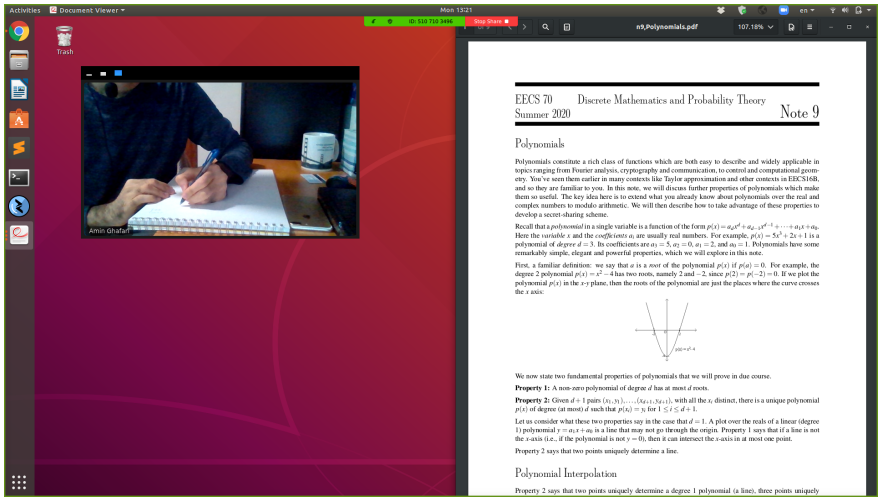


Figure 2: Demonstration of taking your exam. Your setup should look like this while you are taking the exam.

7. Your microphone should be on at all times. We should be able to see the time on your desktop at all times.

8. Write down following the honor code in this state. Then read it out loud. This will ensure you understand the positioning requirement, and have enabled audio.

   I pledge to uphold the university's honor code: to act with honesty, integrity, and respect for others, including their work. By signing, I ensure that all written homework I submit will be in my own words, that I will acknowledge any collaboration or help received,and that I will neither give nor receive help on any examinations.

9. Stop the recording. Upload your video to Google drive and submit the link to the video using this Google Form. You must make sure that the link sharing permissions are set so that we may view the video. Also, please submit your zoom link to this this google form. Afterwards, please write down the magic words from each google form to indicate you have down so.

Link for policy:
https://docs.google.com/document/d/1kj83rPvxgoLuaafkJLcZsmIslWjrhxDujfng4K5oSPI/edit?usp=sharing
Link to upload video:

https://forms.gle/emBA5xzK83wn1uRH8
Link to share zoom link:
https://forms.gle/Co9Rdhx6QV413PFG6

2 Countability Practice

(a) Do $(0,1)$ and $\mathbb{R}_+ = (0,\infty)$ have the same cardinality? If so, either give an explicit bijection (and prove that it is a bijection) or provide an injection from $(0,1)$ to $(0,\infty)$ and an injection from $(0,\infty)$ to $(0,1)$ (so that by Cantor-Bernstein theorem the two sets will have the same cardinality). If not, then prove that they have different cardinalities.

(b) Is the set of strings over the English alphabet countable? (Note that the strings may be arbitrarily long, but each string has finite length. Also the strings need not be real English words.) If so, then provide a method for enumerating the strings. If not, then use a diagonalization argument to show that the set is uncountable.

(c) Consider the previous part, except now the strings are drawn from a countably infinite alphabet $\mathscr{A}$. Does your answer from before change? Make sure to justify your answer.

3 Fixed Points

Consider the problem of determining if a function F has any fixed points; that is, we want to know if there is any input x such that $F(x)$ outputs x. Prove that this problem is undecidable.

4 The Complexity Hierarchy

The complexity hierarchy is a monument to our collective understanding of computation and its limitations. In fact, you may already be familiar with the classes P and NP from CS61B. In this problem, we will focus on decision problems like the Halting Problem, where the output is Yes (True) or No (False), and explore the classes RE, coRE, and R.

(a) A problem is recursively enumerable (RE) if there exists a program P that can print out all the inputs for which the answer is Yes, and no inputs for which the answer is No. The program P can print out a given input multiple times, so long as every input gets printed eventually. The program P can run forever, so long as every input which should be printed is at a finite index in the printed output.

Prove that the Halting Problem belongs in RE. Namely, prove that it is possible to write a program P which:

- runs forever over all possible programs M and inputs x, and prints out strings to the console,
- for every (M,x), if $M(x)$ halts, then P eventually prints out (M,x),
- for every (M,x), if $M(x)$ does NOT halt, then P never prints out (M,x).

In this context, P is called an *enumerator. (Hint: Consider the tail of a dove.)*

(b) An equivalent definition of RE is as follows: A problem belongs in RE if there exists a program P' that will output Yes when given an input x for which the answer is Yes. If the answer is No, then P'(x) may output No or loop forever. As an optional exercise, you should be able to convince yourself that this is indeed an equivalent definition.

Prove that the Halting Problem belongs in RE using this equivalent definition. Namely, prove that it is possible to write a program P' which:

- takes as input a program M and input x.
- if M halts on input x, then P' should print Yes.
- if M does not halt on input x, then P' may output No or loop forever.

In this context, P' is called a *recognizer.*

(c) As you might suspect, a problem is co-recursively enumerable (coRE) if its complement is in RE. The complement of a decision problem A is another problem A' where A'(x) is Yes iff A(x) is No, and A'(x) is No iff A(x) is Yes. State the complement of the Halting Problem.

(d) Finally, a problem belongs in the class R if it is computable, meaning there exists a program P that answers Yes when the answer is Yes, and answers No when the answer is No. By definition then, the problem is a computable function if it is computable.

We know that the TestHalt is not computable, and that the Halting Problem belongs in RE. Prove by contradiction that the Halting Problem cannot belong in coRE.

5 Build-Up Error?

What is wrong with the following "proof"? In addition to finding a counterexample, you should explain what is fundamentally wrong with this approach, and why it demonstrates the danger build-up error.

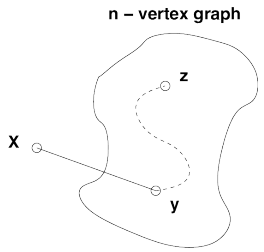
False Claim: If every vertex in an undirected graph has degree at least 1, then the graph is connected.

Proof: We use induction on the number of vertices $n \geq 1$.

Base case: There is only one graph with a single vertex and it has degree 0. Therefore, the base case is vacuously true, since the if-part is false.

Inductive hypothesis: Assume the claim is true for some $n \geq 1$.

Inductive step: We prove the claim is also true for $n+1$. Consider an undirected graph on n vertices in which every vertex has degree at least 1. By the inductive hypothesis, this graph is connected. Now add one more vertex x to obtain a graph on $(n+1)$ vertices, as shown below.



All that remains is to check that there is a path from x to every other vertex z. Since x has degree at least 1, there is an edge from x to some other vertex; call it y. Thus, we can obtain a path from x to z by adjoining the edge $\{x,y\}$ to the path from y to z. This proves the claim for $n+1$.

6 Connectivity

Consider the following claims regarding connectivity:

(a) Prove: If G is a graph with n vertices such that for any two non-adjacent vertices u and v, it holds that $\deg u + \deg v \geq n-1$, then G is connected.

[*Hint:* Show something more specific: for any two non-adjacent vertices u and v, there must be a vertex w such that u and v are both adjacent to w.]

(b) Give an example to show that if the condition $\deg u + \deg v \geq n-1$ is replaced with $\deg u + \deg v \geq n-2$, then G is not necessarily connected.

(c) Prove: For a graph G with n vertices, if the degree of each vertex is at least $n/2$, then G is connected.

(d) Prove: If there are exactly two vertices with odd degrees in a graph, then they must be in the same connected component (meaning, there is a path connecting these two vertices).

[*Hint:* Proof by contradiction.]

7 Triangular Faces

Suppose we have a connected planar graph G with v vertices and e edges such that $e = 3v-6$. Prove that in any planar drawing of G, every face must be a triangle; that is, prove that every face must be incident to exactly three edges of G.

8 Binary Trees

You have seen the recursive definition of binary trees from lecture and from previous classes. Here, we define binary trees in graph theoretic terms as follows (**Note:** here we will modify the definition of leaves slightly for consistency).

- A binary tree of height > 0 is a tree where exactly one vertex, called the **root**, has degree 2, and all other vertices have degrees 1 or 3. Each vertex of degree 1 is called a **leaf**. The **height** h is defined as the maximum length of the path between the root and any leaf.

- A binary tree of height 0 is the graph with a single vertex. The vertex is both a leaf and a root.

(a) Let T be a binary tree of height > 0, and let $h(T)$ denote it's height. Let r be the root in T and u and v be it's neighbors. Show that removing r from T will result in two binary trees, L, R with roots u and v respectively. Also, show that $h(T) = \max(h(L), h(R)) + 1$

(b) Using the graph theoretic definition of binary trees, prove that the number of vertices in a binary tree of height h is at most $2^{h+1} - 1$

(c) Prove that all binary trees with n leaves have $2n - 1$ vertices

9 Euclid's Algorithm

(a) Use Euclid's algorithm from lecture to compute the greatest common divisor of 527 and 323. List the values of x and y of all recursive calls.

(b) Use extended Euclid's algorithm from lecture to compute the multiplicative inverse of 5 mod 27. List the values of x and y and the returned values of all recursive calls.

(c) Find $x \pmod{27}$ if $5x + 26 \equiv 3 \pmod{27}$. You can use the result computed in (b).

(d) Assume a, b, and c are integers and $c > 0$. Prove or disprove: If a has no multiplicative inverse mod c, then $ax \equiv b \pmod{c}$ has no solution.

10 GCD Proof

Let n, x be positive integers. Prove that x has a multiplicative inverse modulo n if and only if $\gcd(n, x) = 1$. (Hint: Remember an iff needs to be proven both directions. The gcd cannot be 0 or negative.)

11 Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student! We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

1. **What sources (if any) did you use as you worked through the homework?**
2. **If you worked with someone on this homework, who did you work with?** List names and student ID's. (In case of homework party, you can also just describe the group.)

3. **How did you work on this homework?** (For example, *I first worked by myself for 2 hours, but got stuck on problem 3, so I went to office hours. Then I went to homework party for a few hours, where I finished the homework.*)

4. **Roughly how many total hours did you work on this homework?**