COS 330: Great Ideas in Theoretical Computer Science

Fall 2026

Precept 1

My Introduction to 330

PROBLEM-SOLVING SESSION

Welcome to COS 330!

In this precept you will learn the format of 330 precepts, and at the same time practice solving some problems and writing proofs.

Each precept will be one of two types. A **Problem-Solving Session** (like today's) gives you problems to work on, either on your own or in small groups; solutions are shared after precept. A **Recap Session** is a short lecture recalling material you should already have seen (with potentially some extras), meant to help you prepare for the following week's lectures.

Since this is the first precept, we'll start by getting to know your TA before turning to the problems below.

Problem 1 comes with a guided walkthrough, so you can see the kind of reasoning we expect from you on problems like these. Problem 2 is yours to work through: it carries a few notes to think with, but no walkthrough.

Problem 1

A graph is *bipartite* if its nodes can be split into two disjoint sets L and R such that every edge has one endpoint in L and the other in R.[1] The *degree* of a node is the number of edges incident to it, and for a set of nodes S we write $N(S)$ for the set of nodes with an edge to some node in S. A *matching* is a set of edges no two of which share an endpoint, and a matching is *perfect* if every node of the graph is in exactly one of its edges.

You will need the following classical theorem, which you may use without proof (we do not assume you have seen it before):

Hall's Marriage Theorem. *Let G be a bipartite graph with $|L| = |R|$. Then G has a perfect matching if and only if $|N(S)| \geq |S|$ for every set $S \subseteq L$ of nodes on the left.*

Let G be a bipartite graph with n nodes on each side. Prove that if every node has degree $\geq n/2$, then G has a perfect matching.

Step One: Understand what the question is asking. This might seem simple, but is well worth spending time on; it's all too easy to try to prove something hard (or even false) because we

[1] Some things can be defined in multiple equivalent ways: for example, a bipartite graph and a 2-colorable graph (where we can color each node, say, red and blue so no edge connects two red or two blue nodes) are the same thing.

misinterpreted a definition. Before reading on, make sure you can state precisely, in your own words:

- what a *graph* is;
- what it means for a graph to be *bipartite*;
- what a *matching* is;
- what makes a matching *perfect*;
- what the *degree* of a node is;
- what $N(S)$ means;
- what Hall's Marriage Theorem promises, and what you would have to check in order to apply it.

It's absolutely fine not to know any of this by heart.

> **Aligning Expectations**
>
> The only things you are expected to know coming into this problem are the basic definitions of a graph (nodes, edges, degree). Everything else – including Hall's Marriage Theorem – is stated in the problem itself. We will do this often, even with material we cover in class, and we will do it on exams too: when a problem needs a definition or an outside theorem, we give it to you. On PSets, you are of course also free to look definitions up online.
>
> The goal of the course isn't for you to memorize definitions. It is to learn how to read, understand, and apply definitions in context – a skill that outlasts any particular definition. Notice, along those lines, how little of the text above is the actual question: the first paragraph is definitions, the second is a theorem you may use, and the question itself is a single sentence.

Step Two: Try Stuff! Understanding exactly what you're supposed to prove is a (significant) part of a solution, but from then on there's no sure-fire strategy guaranteed to work. General tips are: try to construct a counterexample and see where you get stuck, or working through small examples (or both).

Let's consider the smallest non-trivial example of the problem we're trying to solve: with $n = 2$, call $L = \{u, v\}$ and $R = \{x, w\}$. The hypothesis of the statement says every node has $\geq n/2 = 1$ neighbor; that is, u is connected to x or w (or both), as is v (and x, w are each connected to one or both of u, v).

What would a "bad" G look like? Well, maybe both u and v are connected to w but not to x; this satisfies the degree condition for u, v and w but violates it for x. Of course, since we're trying to prove a true statement,[2] there's no such thing as a "bad" G; but it's a useful exercise to try to come up with graphs that "almost break" the statement.

OK, since we're given Hall's Marriage Theorem, let's use it. There are three sets $S \subseteq L$ to consider:

[2] We'll never ask you to prove something false – at least not on purpose!

1. $S = \{u\}$;
2. $S = \{v\}$; and
3. $S = \{u, v\}$.[3]

Since u and v both have at least one neighbor, that covers the cases $S = \{u\}$ and $S = \{v\}$. But how about $S = \{u, v\}$? Well, the only way for $N(S)$ to be smaller than S is if u and v are both (only) connected to one node on R, say, x. But as we just saw (while trying to come up with a "bad" G), that would violate the degree condition on w, so $N(S) = R$ and the hypothesis for Hall's Marriage Theorem is satisfied; so G does have a perfect matching, and we solved the case $n = 2$.

Step Three: Generalize. Note that **this is not a full solution yet, but it makes a lot of progress**! And the intuition we gathered makes coming up with a full proof much more manageable. Indeed, we can follow the same blueprint of a *proof by contradiction*: show that for small S the degree condition immediately implies $|N(S)| \geq |S|$, then show that $|N(S)| < |S|$ for large S would violate the hypothesis. That is:

1. if $|S| \leq n/2$, any $v \in S$ satisfies $|N(\{v\})| \geq n/2$. Then $|N(S)| \geq |N(\{v\})| \geq n/2 \geq |S|$;
2. if $|S| > n/2$ and $|N(S)| < |S|$, any $w \in L$ that is not connected to S has at most $|R| - |S| < n/2$ neighbors, which contradicts the hypothesis.

Comparing the general case with the $n = 2$ one, you may see they're very similar: we took inspiration from a small example to prove the general case, and it worked!

Now, let's look at another approach to the problem (that will fail – your task is to figure out why). There are many proof strategies you can try, contradiction being only one of them. You can try to use induction and/or prove the contrapositive, for example. Here's a seemingly promising "proof" by induction on the number n of vertices, which doesn't use Hall's Marriage Theorem:

> In the base case, $n = 1$, the single node in L is connected to $\geq 1/2$ nodes in R. Since the number of nodes is an integer, there's one edge from L to R, which is itself a perfect matching.
>
> For the inductive step, take any node $u \in L$ and any edge $\{u, w\}$ with $w \in R$. (There is at least one, since the degree of u is $\geq n/2 \geq 1$.) Consider the graph G' with $L' = L \setminus \{u\}$, $R' = R \setminus \{w\}$ and all the edges from L' to R' that exist in G.
>
> We only removed one node from R, so the degree of all nodes in L' is $\geq \frac{n}{2} - 1 \geq \frac{n-1}{2}$. The same argument shows the degree of nodes in R' is also $\geq \frac{n-1}{2}$. By the inductive hypothesis, G' has a perfect matching; adding the edge $\{u, w\}$ to this matching yields a perfect matching for G.

What is wrong with the "proof" above? Are there special cases when it is correct?

[3] Can you see why there's no need to consider $S = \varnothing$?

Problem 2

Let I be a set of n intervals described by $\{[s_i, e_i]\}_{i=1}^{n}$, where s_i and e_i are integers representing the start and end times of the i-th interval. Assume that $s_i < e_i$ for all i. We say that two intervals overlap if they share any point. The maximum cardinality of non-overlapping intervals is the largest number of intervals we can select from I so that no two of them overlap.

(a) Consider the following greedy algorithm to find a set of non-overlapping intervals, which we'll call the `start-time` algorithm:

- Sort the intervals in increasing order of their start times.
- Initialize an empty set S to store the selected intervals.
- Iterate through the sorted intervals, and for each interval $[s_i, e_i]$, if it does not overlap with any interval already in S, add it to S.

Show that this algorithm does not always find the maximum cardinality of non-overlapping intervals.

Problem Solving Tips

You are asked to refute a "for every input" claim, and for that a single input where the algorithm fails is a complete proof – you do not need to characterize when it fails, or by how much.

As in Problem 1, start as small as you can. Try to build an instance out of two or three intervals, and ask what the `start-time` rule is bad at: which interval does it commit to first, and what can that one commitment cost you? Once you have a candidate, remember that the counterexample isn't finished until you have said both what the algorithm returns on it *and* what the true maximum is.

(b) Consider the following greedy algorithm to find a set of non-overlapping intervals, which we'll call the `end-time` algorithm:

- Sort the intervals in increasing order of their *end* times.
- Initialize an empty set S to store the selected intervals.
- Iterate through the sorted intervals, and for each interval $[s_i, e_i]$, if it does not overlap with any interval already in S, add it to S.

Prove the following three things:

1. The `end-time` algorithm always returns a set of non-overlapping intervals.
2. There is no set of non-overlapping intervals larger than the set returned by the `end-time` algorithm.
3. The `end-time` algorithm can be implemented in time $O(n \log n)$.

Problem Solving Tips

This part is really three separate proofs. Break it down and tackle them one at a time – they need quite different kinds of argument:

- The first should follow almost immediately from how the algorithm is written. Short does not mean hand-waved: say precisely which step of the algorithm guarantees it.
- The second is the hardest one. It is a claim about *every* set of non-overlapping intervals, so you cannot argue by inspecting just one.
- The third is a bookkeeping argument: name the cost of each step, and say what you need to store so that the overlap check takes constant time.