CS109B Notes for Lecture 4/17/95

NP-Complete Problems

We have met some problems that have "easy" solutions; they have algorithms that run in time that is polynomial in the *size* of the graph, the parameter m.

- Examples: testing for cycles, finding MWST's or CC's, finding the shortest path between two nodes, testing whether a graph is bipartite.
  - None takes more than $O(m \log n)$, which is surely less than the polynomial $O(m^2)$.
- On the other hand, some problems seem to take time that is exponential in the size of the graph, 2^n or worse.
  - Examples include TSP, tripartiteness, many others, such as the following:

Cliques

A complete subgraph of an undirected graph, i.e., a set of nodes of some graph that have every possible edge.

- The *clique problem*: given a graph G and an integer k, is there a clique of at least k nodes?

Independent Set

Subset S of the nodes of an undirected graph such that there is no edge between two members of S.

- The *independent set problem*: given a graph G and an integer k, is there an independent set with at least k nodes?
- Application: Let nodes = courses. Edge $\{u, v\}$ means that courses u and v have at least one student in common. Here, independent set = set of courses whose finals can be given at the same time.

Colorability

An undirected graph is k-colorable if we can assign one of k colors to each node so that no edge has both ends colored the same.

- The *chromatic number* of a graph = the least number k such that it is k-colorable.
- The *coloring problem*: given a graph G and an integer k, is G k-colorable?

Example: K_n is n-colorable, and its chromatic number is n.

- It is also k-colorable for any $k \geq n$.
  - Note that you do not have to use all k colors in a k-coloring.

Example: "Bipartite" is a synonym for "2-colorable," and "tripartite" is a synonym for "3-colorable."

Checking Solutions Can Be Easier Than Finding Them

Each of the above 3 problems have the interesting property that, while it is hard to find solutions, e.g., "find a clique of k nodes," it is easy (polynomial time to be precise) to check that a proposed solution really is a solution.

- Check a proposed clique by checking for the existence of the $\binom{k}{2}$ edges among the k nodes.
- Check for an independent set by checking for the nonexistence of any edge between two nodes in the proposed set.
- Check a proposed coloring by checking each edge in the graph and confirming that the ends are colored differently.

The Class of Problems NP

Problems such as the above whose solutions can be checked in polynomial time are called *NP* (nondeterministic, polynomial) problems.

- Some, e.g., shortest paths, are truly easy; they can be solved as well as checked in polynomial time.
- Others, such as clique, independent set, or colorability, appear not to be solvable in polynomial time.
- While there is no proof that they cannot be solved in polynomial time, we have the next best thing: a theory that says many of these problems are *as hard as any in NP.*
  - These are called *NP-complete problems.*
- If one NP-complete problem were solvable in polynomial time, then all would be.
  - Since the NP-complete problems include many that have been worked on for centuries, there is strong evidence that all NP-complete problems really require exponential time to solve.
- See p. 673, FCS for a discussion of NP-completeness and the first-known NP-complete problem (tautology for propositional logic).

Reductions

The way a problem is proved NP-complete is to "reduce" a known NP-complete problem to it.

- The first NP-complete problem, tautology, was proven in another manner.
- We *reduce* problem A to problem B by devising a solution for A that uses only a polynomial amount of time plus calls to a subroutine that solves B.

Example: Clique and Independent Set can be reduced to one another easily.

- *Reduce Clique to Independent Set.* Given a graph G and integer k, suppose we want to know if there is a clique of size k in G.

- Construct graph H with the same set of nodes as G and an edge $\{u, v\}$ iff G does *not* have edge $\{u, v\}$.
- An independent set in H is a clique in G.
- Use the "independent set subroutine" to tell whether H has an independent set of k nodes.
- Give the same answer that the subroutine gives.

- *Reduce Independent Set to Clique.* Given G and k, suppose we want to know if there is an independent set of size k.
  - Construct H again, and use clique subroutine to tell if H has a clique of size k.
  - Say G has an independent set of size k iff the subroutine says H has a clique of size k.

Class Problem

Here are two more problems that happen to be NP-complete.

- The *Node Cover* problem: given undirected graph G and integer k, is there a set C of k nodes such that each edge of G has at least one end at a node in C.
  - C is called a *node cover*.
- The *Set Cover* problem: given a set of subsets of $\{1, 2, \ldots, n\}$ and an integer k, determine whether there is a set of k subsets such that each integer between 1 and n is in at least one of the k subsets.
  - The *size* for Set Cover is the sum of the sizes of all the subsets.

Find a reduction of Node Cover to Set Cover. The other way is possible too, but much harder.