NP-completeness

Outline

- Examples of Easy vs. Hard problems
  - Euler circuit vs. Hamiltonian circuit
  - Shortest Path vs. Longest Path
  - 2-pairs sum vs. general Subset Sum
- Reducing one problem to another
  - Clique to Vertex Cover
  - Hamiltonian Circuit to TSP
  - TSP to Longest Simple Path
- NP & NP-completeness

When is a problem easy?

- We've seen some "easy" graph problems:
  - Graph search
  - Shortest-path
  - Minimum Spanning Tree
- Not easy for us to come up with, but easy for the computer, once we know algorithm.

When is a problem hard?

- Almost everything we've seen in class has had a near linear time algorithm
- But of course, computers can't solve *every* problem quickly.
- In fact, there are perfectly reasonable sounding problems that no computer could ever solve in *any* amount of time.

Today

- We'll see 3 pairs of similar problems, one version easy, one version hard.
- The lesson is that seemingly mild changes in problem statement can have a dramatic effect on algorithm speed.

Euler Circuits

- A cycle that passes through every edge exactly once.
- Give example graph (square with X through it.)

Hamiltonian Circuit

- A cycle that passes through every vertex exactly once.
- Give example graph

Finding an Eulerian Circuit

- Very simple criteria: If every vertex has even degree, then there is an Eulerian circuit.
- Reason: If a node has even degree, then one edge used to get to a node, and one edge used to get out. Never get stuck.

Finding a Hamiltonian Circuit

- Nothing to do but enumerate all paths and see if any are Hamiltonian.
- How many paths? *Draw example from box graph.*
- Can think of all paths as a tree. Branching factor approximated by average degree *d*. Then d^N leaves (paths). Exponential. Recall exponential curves from first lecture.

Shortest vs. Longest Path

- Finding the shortest path is easy--that is, we know an efficient algorithm. Namely DFS or BFS.
- How do we find the longest path?

Longest Path

- Again, no choice but to enumerate all paths.
- Q: Why doesn't DFS work?
  - A node is visited only once, therefore only one path through each node is considered. But as we saw, there could be exponentially many paths. DFS is exploring only one per node.

Subset Sum

- We saw 4 number sum in homework:
- Given a list of N integers and target k, are there 4 numbers that sum to k?
- General Subset Sum: Given N integers and a target k, is there some subset of integers that sum to k?

Solving Subset Sum

- Only thing to do is try every possible combination.
- How many possible subset are there of N integers?
  - 2^N. So again, exponential in input size.
- For 4 numbers there are N choose 4 possible subsets to try. Approx. N^4.

Outline: Day 2

- We've seen that there are a bunch of problems that seem to be hard.
- Today we'll see how these problems relate to one another.
- Def: P_1 is reducible to P_2 if there is a conversion from an instance X of P_1 to an instance Y of P_2 such that P_1 is yes for X iff P_2 is yes for Y.

Clique to Vertex Cover

- We can reduce Clique to Vertex Cover.
- Given an input (G, k) to Clique:
  - Build graph G complement
  - Let k' = n – k
- Vertex Cover is "as hard as" Clique.

➔

- If G has a k Clique then G' has a k' cover:
  - Let C be the clique of size k. Let the cover be V-C. Then clearly every edge outside C is covered, and in G' there are no edges in C.
  - Size is n-k

←

- If G' has a k' cover then G has a k Clique:
  - Let D be a cover in G' of size k'. Then there are no edges in V-D, since otherwise they wouldn't be covered. Therefore, V-D is a clique in G.
  - Size of clique is n-k'.

TSP

- Travelling Salesman Problem:
  - Given complete weighted graph G, integer k.
  - Is there a cycle that visits all vertices with cost <= k?
- One of the canonical problems.
- Note difference from Hamiltonian cycle: graph is complete, and we care about weight.

Hamiltonian Cycle to TSP

- We can reduce Hamiltonian Cycle to TSP.
- Given graph G=(V, E):
  - Construct complete graph G' on N vertices with edge weights: 1 if (u, v) in E, 2 otherwise.
  - Let k = N.
- *Do example with N=5.*
- *TSP is "as hard as" Hamiltonian cycle.*

Proof

- If G has a Hamiltonian Cycle then G' has a tour of weight N.
  - Obvious.
- If G' has a tour of weight N, then G has a Hamiltonian Cycle.
  - Obvious.

Ham. Cycle to longest path

- Recall, Longest Path: Given directed graph G, start node s, and integer k. Is there a simple path from s of length >= k?
- We'll use Directed Hamiltonian Cycle.

The reduction

- Given a directed graph G.
  - Pick any node as start vertex s.
  - Create a new node t. For every edge (u, s), add an edge (u, t). Let k = N.
- Draw example
- Longest Path is "as hard as" Ham. Cycle

Proof

- If G has a Ham. Cycle, then G' has a path of length k from s.
  - Follow the cycle starting at s, at the last step go to t instead of s.
- If G' has a path of length k from s, then G has a Ham. Cycle.
  - Path must have hit every node exactly once, and last step in path could have formed cycle in G.

NP-completeness

- We've seen that there are seemingly hard problems. That's kind of interesting.
- The really interesting part: A large class of these are equivalent. Solving one would give a solution for all of them!

- The pairs I picked weren't important. There is a large class of problems, called NP-complete, such that *any* one can be reduced to *any* other.
- So given an algorithm for any NP-complete problem, all the others can be solved.
- Conversely, if we can prove there is no efficient algorithm for one, then there are no efficient algorithms for any.

What's NP-complete

- Satisfiability of logic formulas
- All sorts of constraint problems
- All sorts of graph problems
- Not an overstatement to say that every area of computer science comes up against NP-complete problems.

What do we do about it?

- Approximation Algorithm:
  - Can we get an efficient algorithm that guarantees something close to optimal?
- Heuristics:
  - Can we get something that seems to work well most of the time?
- Restrictions:
  - Longest Path is easy on trees, for example. Many hard problems are easy for restricted inputs.