The Clique Problem

A *clique* in a graph is a completely connected subgraph

- a set of nodes such that every pair is connected by an edge

The CLIQUE problem: Does a given graph have a clique of size k?

- CLIQUE $= \{(G, k) : G$ has a clique of size $k\}$.

Theorem: CLIQUE is NP-complete.

Proof: Clearly, CLIQUE is in NP: just guess the nodes in the clique and verify that they're all connected to each other.

To show CLIQUE is NP-hard, reduce 3-CNF to CLIQUE:

- Given a 3-CNF formula φ with k clauses, find a graph G such that G has a clique of size k (i.e. $(G, k) \in$ CLIQUE) iff φ is satisfiable.

Idea of proof:
Suppose $\varphi = C_1 \wedge \ldots \wedge C_k$;

- $C_i = l_{1k} \vee l_{2k} \vee l_{3k}$, where l_{ij} is a literal

Construct a graph $G_\varphi = (V_\varphi, E_\varphi)$ as follows:

- put a vertex v_{ij} in V_φ for each literal l_{ij}, $i = 1, 2, 3$, $j = 1, \ldots, k$
  - That means that V_φ has $3k$ vertices
- Put an edge between v_{ij} and $v_{i'j'}$ if
  - $j \neq j'$ (so that l_{ij} and $l_{i'j'}$ are indifferent clauses)
  - l_{ij} is not equivalent to $\neg l_{i'j'}$

Claim: φ is satisfiable iff G_φ has a k-clique

- If φ is satisfiable, there is a truth assignment such that for each j, (at least) one of l_{1j}, l_{2j}, and l_{3j} is true.
  - Pick one: the set of literals you picked forms a k-clique
- if there is a k-clique, define a truth assignment that makes each literal in the k-clique true
  - This gives an assignment that satisfies φ

Vertex Cover

A *vertex cover* of a graph $G = (V, E)$ is a set V' of vertices that *covers* the edges:

- Each edge in E is incident to at least one vertex in V'
- If $(i, j) \in E$, then $i \in V'$ or $j \in V'$

We are typically interested in finding a *minimum* vertex cover:

- Mimimum number of agents needed to watch all the roads.
- Used in computational biochemistry to exclude conflicts in samples:
  - vertices in graph are observations
  - put an edge between two vertices when they conflict
  - a minimum vertex cover is a minimum set of conflicting observations.

The decision problem version:

VERTEX-COVER $= \{(G, k) : G$ has a vertex cover with k vertices$\}$

Theorem: The vertex-cover problem is NP-complete.

Clearly vertex cover is in NP:

- Given (G, k), guess a set V' with $|V'| = k$ and verify that it covers the edges in G.

To show vertex-cover is NP-hard, we reduce the clique problem to it.
Given a graph $G = (V, E)$, want to know if it has a clique of size k:

- Let $\overline{G} = (V, \overline{E})$
- $\overline{E} = \{(u, v) : (u, v) \notin E\}$

Claim: G has a clique of size k iff $\overline{G}$ has a vertex cover of size $|V| - k$.

- V' is a clique in G iff $V - V'$ is a vertex cover for $\overline{G}$.
    - If $V - V'$ is a vertex cover of $\overline{E}$ and $u, v \in V'$ then $(u, v) \in E$
        - if $(u, v) \in \overline{E}$, then $u \in V - V'$ or $v \in V - V'$
        - That means V' is a clique
    - If V' is a clique and $(u, v) \in \overline{E}$, then $u \in V - V'$ or $v \in V - V'$, so $V - V'$ is a cover of $\overline{E}$.

Approximation Algorithms

VERTEX-COVER is NP-complete.

- Can't expect to find a PTIME algorithm to find the smallest vertex cover.

There is a PTIME algorithm that finds an "approximate" vertex cover.

- At most $2\times$ size of smallest vertex cover.

Idea: use a *greedy algorithm*.

- Keep a list of uncovered edges E' and a tentative cover V'
- Initially $E' = E$, $V' = \emptyset$
- Pick an uncovered edge $(u, v) \in E'$ at random
  - add u, v to V'
  - remove from E' all edges covered by u, v
- Continue until E' is empty

This works:

- V' must be a vertex cover at the end.
- If V'' is any vertex cover, for every pair u, v included in V', at least one of u, v must be in V''

    - Otherwise it wouldn't cover (u, v).
- Therefore, $|V''| \leq 2|V'|$.

Can we do better?

- Is there a PTIME algorithm for vertex cover with an approximation factor of better than 2?
- It's NP-complete to approximate it better than 1.1666.

Hamiltonian Cycle

HAM-CYCLE = $\{G : G$ has a Hamiltonian cycle$\}$

Theorem: HAM-CYCLE is NP-complete.

Proof: Clearly it is in NP.

To show that it is NP-hard, we reduce 3-CNF to it.

- This is tricky ...
- See text.

Hamilton Cycle vs. Hamilton Path

Both the Hamiltonian path problem and the Hamiltonian cycle problem are NP-complete. It's easy to reduce one to the other:

Reducing Hamilton cycle to Hamiltonian Path:

- Given a graph $G = (V, E)$; choose a vertex $v_0 \in V$
- Define $G' = (V', E')$ as follows:
  - $V' = V \cup$ two special new vertices: x and x'
  - $E' = E \cup$ an edge (x, v_0) from x to v_0 and edges (v', x') for all v' such that $(v', v_0) \in E$.

G has a Hamiltonian cycle iff G' has a Hamiltonian path:

- If G has a cycle, without loss of generality, it is of the form $v_0, \ldots, v_n$, where $v_n = v_0$.
  - $x, v_0, \ldots, v_{n_1}, x'$ is a Hamiltonian path in G'.
- If G' has a Hamiltonian path, it must start with x and end with x', say $x, v_0, \ldots, v_{n-1}, x'$.
  - $v_0, \ldots, v_{n-1}, v_0$ is a Hamiltonian cycle in G.

Reducing Hamiltonian path to Hamiltonian cycle:

- Given $G = (V, E)$, define $G' = (V', E')$ as follows:
  - $V' = V \cup$ one special vertex x.
  - $E' = E \cup (x, v)$ and (v, x) for all $v \in V$
- If $v_0, \ldots, v_n$ is a Hamiltonian path in G iff $(x, v_0, \ldots, v_n, x)$ is a Hamiltonian cycle in G'.

Traveling Salesman Problem

This is a weighted version of HAM-CYCLE.

- Each edge has a weight.
- Is there a Hamiltonian cycle with weight $\leq k$?
- Think of a salesman going from city to city where the weight is the travel time.
- In text, it is assumed that the graph is complete.
    - Can go anywhere, at a cost.

TSP $= \{(G, k)$: there is a Hamiltonian cycle in G of weight $\leq k\}$

Theorem: TSP is NP-complete.
Proof: Clearly in NP. To show it is NP-hard, reduce Hamiltonian path to it.

- Given a graph G with n vertices, give each edge weight 1.
- There is a Hamiltonian cycle in G iff we can solve TSP for (G, n).

There are PTIME algorithms for approximating TSP to within $1 + \epsilon$, for any fixed ϵ!

- The polynomial has degree roughly $1/\epsilon$.

Subset Sum

Given a set of S of natural numbers and a target sum $t \in N$, is there a subset S' whose elements sum to t?

- $S = \{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}$, $t = 3754$.
- Answer: Yes – take $S' = \{1, 16, 64, 256, 1040, 1093, 1284\}$

This is the problem of making change, given a collection of coins.

Theorem: Subset-sum is NP-complete

Proof: Clearly in NP.

For NP-hardness, reduce Hamilton cycle to subset sum.

- Suppose that there are 5 vertices, $1, \ldots, 5$
- Represent an edge $(2, 4)$ as the number 1011100010.
  - The first 5 digits 10111 have a 1 everywhere but at 2
  - The last 5 digits 00010 have a 1 only at the 4.

- The fact that (2,4) comes before (4,3) in a path will be encoded by the the 11101 that starts (4,3) matching up with the 00010 in (2,4).
- A careful proof is written up and available on the course web site.

0-1 Knapsack

Theorem: The 0-1 Knapsack problem is NP-complete.

Clearly 0-1 knapsack is in NP:

- Guess a set of items, see if their total weight is $\leq W$ and total value is $\geq V$.

To show that 0-1 Knapsack is NP-hard, reduce subset sum to it.

Given set S and target t:

- If $j \in S$, have item v_j with weight and value j
- Let $W = V = t$.
  - You can get a set of items of weight at most W with value at least V iff there is a subset of S that sums to t.

Coverage of Final

Final is on Thursday, May 17, 3-5:30, Hollister 362.

It includes:

- Everything that was on the prelim.
- Shortest paths
  - Dijkstra, Bellman-Ford
- Minimum spanning trees
  - Kruskal, Prim
- Flow networks.
- Dynamic programming
- Greedy algorithms
- NP-completeness
  - PTIME
  - Reductions
  - Languages (L^*)

Review session on Wednesday, May 16, 4–6 PM, in Upson 207.

- Office hours continue as usual, except Bo Pan will be covering mine for on Tuesday, May 15, and Bo's Wednesday office hours will be 1-2, not 10-11, and in her office.